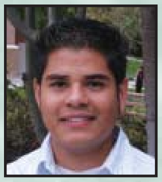
Security Requirements Engineering: A Survey



Jose Romero-Mariona
University of California, Irvine
jromerom@uci.edu

Hadar Ziv
University of California, Irvine
ziv@ics.uci.edu

Debra J. Richardson
University of California, Irvine
djr@ics.uci.edu

August 2008

ISR Technical Report # UCI-ISR-08-2

Security Requirements Engineering: A Survey

Jose Romero-Mariona, Hadar Ziv, and Debra J. Richardson
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425
{jromerom, ziv, djr} @ics.uci.edu

ISR Technical Report # UCI-ISR-08-2

Abstract

Security has become a primary and prevalent concern for software systems. The past decade has witnessed a tremendous increase in not only the sheer number of attacks but also the ease with which attacks can be performed on systems. We believe that in order to protect a system against harm (intended or not), attention must be given to its requirements. Similar to other system properties and quality attributes, security must be considered from inception, in other words starting with requirements. Security is a nonfunctional requirement (NFR) that is increasingly critical in its importance, unique in its requirements, yet must still be integrated with all other functional and non-functional requirements and mapped into successful architectures, designs, and implementation. Similar to other nonfunctional requirements, the unique nature and demands of security make it difficult and often ineffective to specify security concerns using "general purpose" requirements methods. As a result, several original and derived approaches to security requirements engineering have recently been proposed.

In the following survey we explore a variety of approaches for engineering security-specific requirements. For the purposes of this survey, we decompose security requirements engineering into five more manageable phases, namely, security requirements elicitation, security requirements analysis, security requirements specification, security requirements management, and later stages support for security requirements. We have developed an evaluation framework that focuses on each phase; the evaluation framework is composed of a variety of questions and response criteria designed in order to probe how well existing approaches support each specific security requirements engineering phase.

We survey a total of 12 approaches; there are 6 approaches that have been derived from other approaches in order to address security, and there are 6 approaches that have been developed specifically for security requirements. We apply our evaluation framework to each of the 12 approaches and rank their responses based on a "star count" system. The stars possible for each response range from 0 to 3. 0 stars indicate no support and 3 stars indicate the maximum level of support for that question. Based on the results of the survey, we provide a variety of observations and propose recommendations for improving security requirements engineering. With our survey, we also uncover a variety of areas in security requirements engineering in which there is an evident lack of support; these areas of need will become part of our future work.

Table of Content

1 Introduction

Requirements engineering is the first major stage of software development. It is during this stage, that the customer and developers come to an agreement as to what constitutes the software to be developed. As one might expect, this is a critical stage of development because anything that is (or is not) resolved at this time will be carried down the rest of the software lifecycle. Good requirements engineering is therefore essential for successful system development [Mea06]. Non-functional requirements, and security ones specifically, would benefit tremendously from a proper requirements engineering approach. The following research, surveys several requirements engineering approaches that are geared specifically towards security requirements. The focus of the survey is thus centered on the interaction between requirements engineering and security requirements.

During our survey, we examine a total of 12 approaches to security requirements engineering. In order to make the survey more manageable and our application of the framework more organized, we decomposed security requirements engineering into 5 distinct phases,

1. Security Requirements Elicitation
2. Security Requirements Analysis
3. Security Requirements Specification
4. Security Requirements Management
5. Later Stages Support for Security Requirements

A variety of questions have been designed for each specific phase, in order to investigate how well each approach performs at each one of the security requirements engineering phases. There are a total of 34 questions for the whole survey; each one with specific criteria for the responses expected.

We believe that a system can be better protected when its security aspects are tackled early on, i.e., during the requirements stage of development, and then carried into further stages. The objectives of the survey are the following;

1. Identify the deficiencies and advantages of current security requirements approaches.
2. Use the results from (1) to determine current needs of security requirements engineering.
3. Determine whether certain approaches are beneficial enough for security requirements and thereby worth extending past requirements into architecture, design, and implementation.
4. If no approach is found beneficial enough, the survey could form the basis of the development of a new security requirements engineering approach whose main goal is to maximize the benefits and limit the deficiencies found in existing approaches.

Along with the objectives for our survey, we also have a set of initial expectations that we anticipate our survey to either refute or validate,

1. Most of the approaches surveyed will do very well in the initial 4 phases of security requirements engineering (elicitation, analysis, specification,

management) but there will be a lack of support for later stages of development (i.e. architecture, design, implementation, and testing)

2. Approaches created specifically for security requirements should do better than those that have been adapted from existing ones to address security
3. It is not expected that one specific approach will be extraordinarily better than all the other 11 surveyed

In order for security to be "built-in" a system, a good security requirements engineering approach should be selected for that task. This survey will help in the selection of that approach. The survey provides the reader not only with explanations to the responses given to each question for each approach, but also a comparison from different aspects of all 12 approaches surveyed.

For this survey, we consider requirements engineering to be "the disciplined application of scientific principles and techniques for developing, communicating, and managing requirements" [STEP91]; and believe that while security can be emphasized at various stages in the software lifecycle, the requirements stage is vital.

Security is uniquely complex and challenging among NFRs; as Ian Alexander indicates, "security is unlike all other areas in a specification, as someone is consciously and deliberately trying to break the system" [Ale02]. Security is a NFR that is increasingly critical in its importance, unique in its requirements, yet still must be integrated with all other functional and non-functional requirements and mapped into successful architectures, designs, and implementation [Rom07].

The following are some definitions of what security is (or could be),

- According to Gary McGraw [McG03], "Software security is about understanding software induced security risks and how to manage them."
- Clark Hayden et al. [Sto01], consider security to be a system property. According to them, "security is much more than a set of functions and mechanisms; IT security is a system characteristic as well as a set of mechanisms that span the system both logically and physically."

While the definitions above prove to be useful in certain areas, we found that security, as presented by Redwine et al. from the Software Process Subgroup within the Task Force on Security across the Software Development Lifecycle of the National Cyber Security Summit, is more complete and adequate for the work at hand. Software security has as its primary goals three aspects, the preservation of the confidentiality, integrity, and availability of the information assets and resources that the software creates, stores, processes, or transmits including the executing programs themselves. In this sense, confidentiality preservation refers to the prevention of unauthorized disclosure; integrity preservation is about preventing unauthorized alteration; and availability preservation is about preventing unauthorized destruction or denial of access or service [Red04].

Addressing security in software development is extremely important; particularly because of the effects software security has on society. Today, software security problems are frequent, widespread, and serious. According to Redwine et al. "the number and variety of attacks by persons and malicious software from outside organizations, particularly via the Internet, are increasing rapidly, and the amount and consequences of insider attacks remains serious" [Red04].

Software security is something that must be taken seriously, but according to McDermott [McG03] software security is intrinsically a difficult task. This difficulty arises due to three main reasons,

- Networks are everywhere: Due to the growing connectivity of computers through the Internet, both the number of attack vectors, and the ease with which an attack can be made have also increased. This growth in networked systems just means that there are more software systems to attack and as a consequence greater risks from poor software security practice than in the past.
- Systems are Easily Extensible: An extensible host accepts updates or extensions, referred to sometimes as mobile code, so that the system's functionality can be evolved. Unfortunately, the very nature of extensible systems becomes a two-edged sword, as it makes it hard to prevent software vulnerabilities from slipping in as an unwanted extension.
- System complexity is rising: This increase in complexity, like in many other software fields, makes it difficult to plan for security, as it is an environment that is constantly changing.

2 Motivation

As an example given by Kenneth Olthoff, let's say that we need to specify a secure network as a system able to securely transport data from one place to another; as developers, we tend to specify the transportation of data as a primary function, "relegating security functions to a sort of requirements ghetto" [Olt01]. Furthermore, Piessens et al. argue "software developers, tending to think of functionality in the first place, usually emphasize convenience over security" [Pie01]. This "relegation" and "convenience over security" should not happen when security is at the heart of a system, and in order to change this we must concentrate on the requirements stage.

The main motivation behind this research is not just to point out the fact that security risks keep on escalating daily, but rather to stress that, just as any other system property, security should be dealt with at the beginning of the software lifecycle. We have decided to focus on the requirements phase because security is a system property. Others agree with us, [Sto01]; they tell us that "security is much more than a set of functions and mechanisms; security is a system characteristic as well as a set of mechanisms that span the system both logically and physically." Security needs to be considered a property of the system to be implemented, and as such, it must be dealt with since the inception of the project.

As expressed by Ronald Lewis [Lew02], "the need for security is often realized too late in the development life cycle." He explains that often the need to focus on security is not realized until the implementation stage, and security measures not added until the maintenance stage [Lin97]. This delay in security focus skyrockets the costs of adding it to the project, as does any other feature that could have been incorporated into the project during the requirements stage, but is not. Furthermore, as Lewis states, "a well-secured system has security designed during initiation, not during implementation or maintenance, because service packs and patches applied after implementation can introduce other vulnerabilities, leading to a spiral of patching, fixing, and re-patching." Since we are focusing in the development of a well secure system, it is imperative that we stress security during the requirements phase.

As we have shown, there is a consensus that it is better to secure a system during the requirements phase, but this is often hard to do without the proper support. This is when our survey comes into play, as it will not only inform you of the current approaches available for security requirements engineering, but will also provide observations and recommendations about them.

3 Security Requirements Engineering

Requirements engineering is a notion that has been around for a number of years, and is well established now. Requirements engineering is defined as "the disciplined application of scientific principles and techniques for developing, communicating, and managing requirements" [STEP91]. When it comes to security, the requirements engineering notion behind it is fairly recent; as codesecurely.org suggests, "one of the most ignored parts of a security enhanced software development life cycle is the security requirements engineering process" [CSO06]. It is only recently, that the idea of considering security early in the development of a system has become popular, as traditional requirements engineering is not enough [Fir07] and we believe that requirements engineering can provide great support for ensuring that security is built into a system as opposed to "bootstrapped" to it later on [Lew02]. Security requirements engineering has become interested with developing a variety of approaches for developing software requirements that have security at heart [Fir03].

Security needs arise when stakeholders determine that some resource belonging to a software system, tangible (e.g. money) or intangible (e.g. confidential information), is valuable to the organization. Such resources are called assets [ChF05, ISO99], and stakeholders wish to protect these assets from any damage or attacks. Security requirements engineering is thus focused on the protection of these valuable assets from the requirements perspective.

We believe that security at the requirements stage should be an essential notion for understanding not only how to secure a system, but what needs to be done in order to ensure that the customer is satisfied with the end-result. In order to help us understand better what has been done in the area of security requirements engineering, we decided that it would be important to decompose into smaller phases. Each of these phases considers an important aspect of the requirements engineering stage. Most of the literature investigated pointed out to a variety of activities involved in requirements engineering; these activities ranged from elicitation to verification and maintenance of software requirements and we believe that these same activities can be used to look at security requirements engineering specifically.

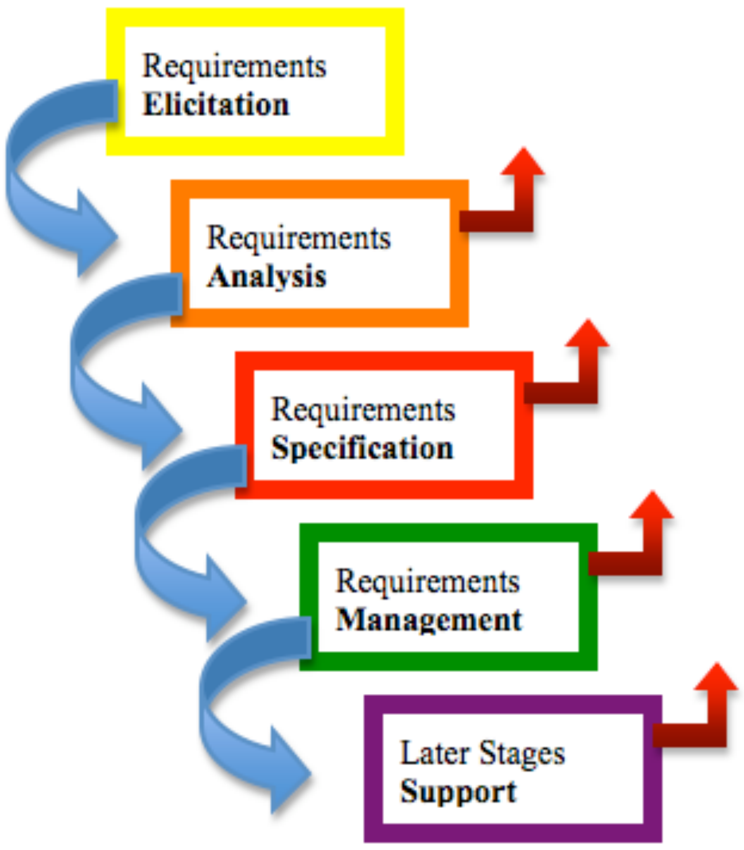


Figure 1. Requirements Engineering Phases Surveyed

For this survey, we are concentrating on 5 main phases of requirements engineering as shown in Figure 1; this decision was made based on the fact that the majority of the work surveyed seemed to agree on the existence/need for at least four of these five different phases. The only phase not mentioned in great detail, is support for not only integrating the security requirements with later stages of the development cycle, but also making them useful. The five phases that we are surveying are:

1. *Security Requirements Elicitation.* This is the initial activity for most of the requirements engineering approaches that we surveyed. This phase is mainly concerned with gathering as much information as possible from a variety of stakeholders including (but not limited to) customers, developers, past documentation, and stakeholders. The purpose of this phase is to have a very good idea as to what (not how) the system at hand is supposed to look and function like.
2. *Security Requirements Analysis.* During this phase, the developers (often with the customers as well) analyze the security requirements that were elicited in order to determine a variety of aspects about them, including their completeness, clarity, and to resolve different aspects like conflicts and ambiguity. This is a very important phase because it helps ensure that the security requirements elicited provide a valid "blue print" of what the customer considers to be a satisfying system.
3. *Security Requirements Specification.* Once the security requirements have been analyzed, it is important to record them in order to make them "official." This is where specification comes into play. During this phase the development team

organizes the security requirements in a way that ensures their recording will be clear, consistent, and traceable, just to mention a few of the characteristics sought after in a security requirements specification document. This phase is extremely important because oftentimes the document produced during specification is what the rest of the development stages will be based upon.

4. *Security Requirements Management.* Similar to the maintenance of a software system, its security requirements must also be properly maintained. Some of the most important maintenance tasks that we consider during this phase include the updating of the security requirements as well as the degree of evolution support that the approach provides.
5. *Later Stages Support of Security Requirements.* While this last phase is not very popular in the literature surveyed (popular in the term that some papers mention it, but rarely any provide support for it) we consider that it is possibly the most important of all the five phases discussed. The importance of this phase lies in the fact that there is a lot of effort poured into eliciting, analyzing, specifying, and maintaining the security requirements but there is not enough support for either integrating the security requirements at later stages of development nor making them useful at these stages. At the heart of this phase is the ability of any given approach to provide support and/or guidance for not only easily but also effectively integrating the security requirements with architectures, design, implementation, and testing of the system. As mentioned above, the effort put into developing "good" security requirements should go beyond requirements themselves, their quality is truly tested when they are used at later stages of development.

4 Related Work

This survey allowed us to observe that while there are some sources that present information on requirements engineering as well as security requirements, information that *discusses and compares* various approaches to security requirements engineering is very limited, almost non-existent. We will discuss some of these sources, and have divided related work into those that focus on requirements engineering and those that focus on security requirements

4.1 Requirements Engineering

Requirements engineering is "the disciplined application of scientific principles and techniques for developing, communicating, and managing requirements" [STEP91]. Similarly, Loucopoulos and Champion define requirements engineering as "the systematic process of developing requirements through an iterative process of analyzing a problem, documenting the resulting observations, and checking the accuracy of the understanding gained" [Lou89].

When it comes to requirements engineering, there is a variety of taxonomies that have been formulated in order to specify the phases involved in it. Davis [Dav1] points out that requirements engineering can be decomposed into elicitation, solution determination, specification, and maintenance. This is very similar to the phases that we are considering with exception of a specific analysis phase and later stages support.

Dorfman decomposes requirements engineering into 5 phases more comparable to ours, specifically elicitation, analysis, specification, validation/verification, and management [Dor90]. While much closer to our notion of phases, they also fail to mention any support for requirements when it comes to later stages of development.

Easterbrook and Nuseibeh [Eas00] look at requirements engineering as having the following phases eliciting requirements, modeling and analyzing requirements, communicating requirements, agreeing requirements, and evolving requirements. While the descriptions of their categorizations correlate to ours, there is still lack for our last surveyed phase, later stages support.

There are also some papers that attempt to "pinpoint" areas where requirements engineering is lacking support. One of the most recognized papers in this category is "Four Dark Corners of Requirements Engineering" by Zave and Jackson [Zav97]. In this paper, they identify four specific areas in requirements engineering that lack support and propose solutions for them. While the information presented is helpful, the problem is that the paper is over 10 years old. In our survey, we also identify areas in security requirements engineering lacking support, but based on today's available approaches.

4.2 Security Requirements

In our survey we discovered that while there has been work done in the area of security requirements [MoH04, MeF07, Mead04] the breadth and depth of this work is far less detailed than our survey. There are two main categories of related work in security requirements; the first one is proposals of frameworks for developing security requirements. Most of these papers are very focused on a specific approach or method that they are either proposing or evaluating. Moffett and Nuseibeh [Mof03] propose a framework for integrating concepts from requirements engineering and security engineering. They take concepts from each one of the disciplines in order to shape the framework in which security requirements should operate. While it provides good information on the context of security requirements, it does not go beyond proposing the framework, let alone applying it to specific security requirements approaches to evaluate them. Giorgino, Massacci, and Zannone also discuss the importance of providing a context for security requirements and propose a framework, which they then use to motivate their approach to security requirements engineering called Secure-Tropos [GiM05]. Haley et al [HaL07] also propose a framework for eliciting and analyzing security requirements; while the framework is actually applied to a specific case study, there is no comparison made between their approach and others.

The second area of related work when it comes to security requirements are papers that attempt to survey either state of the art approaches to security requirements engineering and/or comparable approaches to those being proposed in the papers. The first paper is a joint effort by the Information Assurance Technology Analysis Center (IATAC) and the Data and Analysis Center for Software (DACS) [IAT07]; they provide a very good survey on the state-of-the-art for approaches that support security requirements engineering. While they provide good information about each approach, they lack any comparison between any of the approaches mentioned. We did come across some papers that provide comparisons between surveyed approaches, but they also lack specific aspects that our survey provides. Mellado et al [MeF06] surveys 8 different methods for security requirements engineering. While similar to what we are doing, they only look at 5 specific questions and only 8 approaches. Our survey looks at a total of 34 questions divided among 5 different phases and 12 approaches in total. Also, the approaches surveyed are analyzed solely on a specific paper per approach, as opposed to our survey which looks at all of the possible papers (as many as could be found) describing each approach, in order to give more context and characteristics to each approach. Tondel et al [Tun08] provide a very good look at 9 different approaches for security requirements engineering, but they do not provide a qualitative analysis of their results. While their survey informs you of the capabilities of the various approaches surveyed, it does not go beyond identifying if there is support for each aspect or not (no notion of how well each aspect is satisfied). In our survey, we rank our analysis on a None, Low, Moderate, or High scale; this helps not only identify support for a specific question, but it tells you how well this support is provided.

5 Survey Method

While we consider that this survey is aimed at security requirements engineering on a big scope, we decided to decompose our evaluation framework based on the five phases of security requirements engineering explained in section 3. This decomposition of the framework into five specific areas allows us to survey security requirements engineering in a more manageable manner. Each one of the framework areas has specific questions concerning each phase; these questions have been developed in order to probe how well each of the methods surveyed performs at each phase. The development of the questions in this survey has been mainly done through aspects found in our literature review that point at important characteristics that security requirements and their approaches should have. Even though we discuss limitations of our framework in section 5.3, we believe that the questions used in our framework suffice for the survey at hand.

5.1 Evaluation Criteria

The evaluation criteria in a survey is very important, because this is what determines what is a “good” or “bad” answer to each question of the framework. There are three main types of criteria used in our survey; the first kind refers to answering the question based on “how well the approach fulfills it?”, the second kind of criteria identifies if an approach has a certain aspect, and the third is a combination of both.

The first kind of criteria can be thought of as a rating system; meaning there are some answers better than others. Throughout our survey, this rating system criteria is predominant; out of the 34 questions surveyed 30 of them expect an answer that “rates” the support of each approach for that question on a scale ranging from None to High. We have represented each type of response based on a star system; figure 2 shows the representation of each response based on symbols. An approach that answers a question with a “High”/★★★ is better/more desirable (as more support is provided) than an approach that answers the same question with only a “Moderate”/★★ for example. We decided to use star-system in order to make it easier for the reader to identify the approaches that did better from a high-level view.

While the second kind of criteria (those that merely identify a specific aspect of the approach) do not provide a rating that can show you which answer is better or more desirable, they are still important. We consider that for our survey it would be important to identify specific aspects about each approach surveyed, and let the reader decide how "good" the aspect identified in each approach is; we do this for 4 of the questions.

Some of the questions expect answers that are a little bit of both of the above kinds. These questions do not only expect to identify if a specific approach fulfills certain aspects, but how well does it do it. For example, one of our questions refers to identifying if the security requirements specification produced by the approach can be used for either validation, verification, or both of the implemented system; in this question the interest is not only on identifying that the security specifications can be used for system validation, but that it would support it moderately /★★. We have included this kind of questions in the group of 30 described above as they also provide a rating.

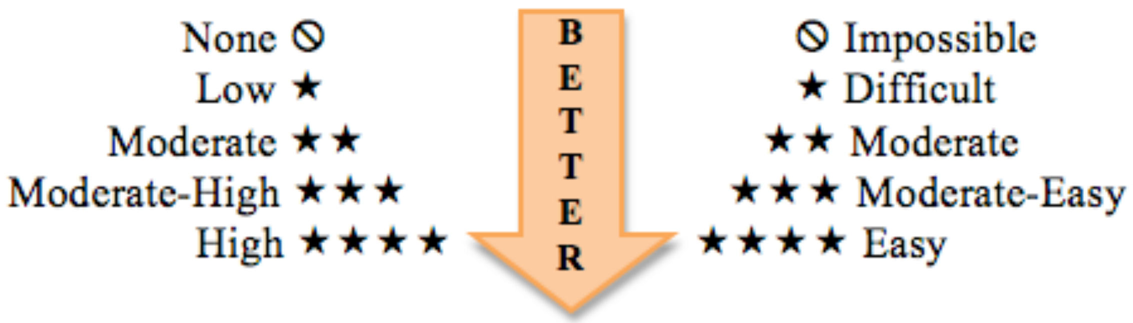


Figure 2. Likert Scale Items

At a high level, the different amount of stars are mutually dependent; for example, a "High"/★★★ ★ answer fulfills not only the criteria for a High answer, but also for a Moderate-High, Moderate, and Low answer. The criteria is determined as follows,

"None"/ ⦸	There is no information regarding any aspects/concepts about the question at hand in any of the sources describing the approach
"Low" /★	Aspects/concepts related to the question at hand are suggested (but not in detail) OR there is enough information to suggest that any support would be possible by the approach
"Moderate" /★★	Aspects/concepts related to the question at hand are explicitly mentioned in any of the sources describing the approach, but it is not a critical component of the approach. Support is described, but no specific measures to operationalize that support are given
"Moderate-High" /★★★	Aspects/concepts related to the question at hand are explicitly mentioned in a majority of the sources describing the approach; they are important aspects to the approach. Support is described and some measures to operationalize that support are described

“High”/★★★ ★	Aspects/concepts related to the question at hand are critical to the approach. Support for the specific question is described across the majority of sources describing the approach. There is evident and extensive support for the question at hand, and measures for achieving this support are described.

5.2 Evaluation Framework Phases

The questions and evaluation criteria have been designed to probe how well each method supports each specific phase. The design of the questions and their placing in a specific phase is mainly based on the important aspects of requirements engineering as identified in our literature survey for each specific phase. Throughout our research we found certain aspects that authors conveyed are important for specific phases of requirements engineering; these findings have been used to develop the questions so that we are surveying meaningful aspects of each approach. For example, the security requirements elicitation phase of our framework has questions that are mostly geared towards issues like level of stakeholder identification, customer involvement level, type of elicitation technique(s) used, etc. While the security requirements specification phase of the framework is mostly interested in probing how consistent, clear, and secure the specifications are.

Below is a breakdown of each specific phase of security requirements engineering that this survey evaluates along with the set of questions and evaluation criteria for that phase.

5.2.1 Security Requirements Elicitation

Requirements elicitation has been defined recently as "the process of identifying needs and bridging the disparities among the involved communities for the purpose of defining and distilling requirements to meet the constraints of these communities" [SEI91]. In this sense, we can see that elicitation goes beyond asking questions to the parties involved; it serves as a front end to the development of the system. Various stakeholders including customers, developers, and end-users are involved with requirements elicitation in a variety of ways, and thus requirements elicitation involves social, communicative issues, and technical issues [Zuk89], [Zah90].

Christel and Kang [Chr92] state that while requirements elicitation can be broken down into the activities of fact-finding, information gathering, and integration. Furthermore, Rzepka decomposes the elicitation process as follows [Rze89]:

1. Identify the relevant parties that are sources of requirements.

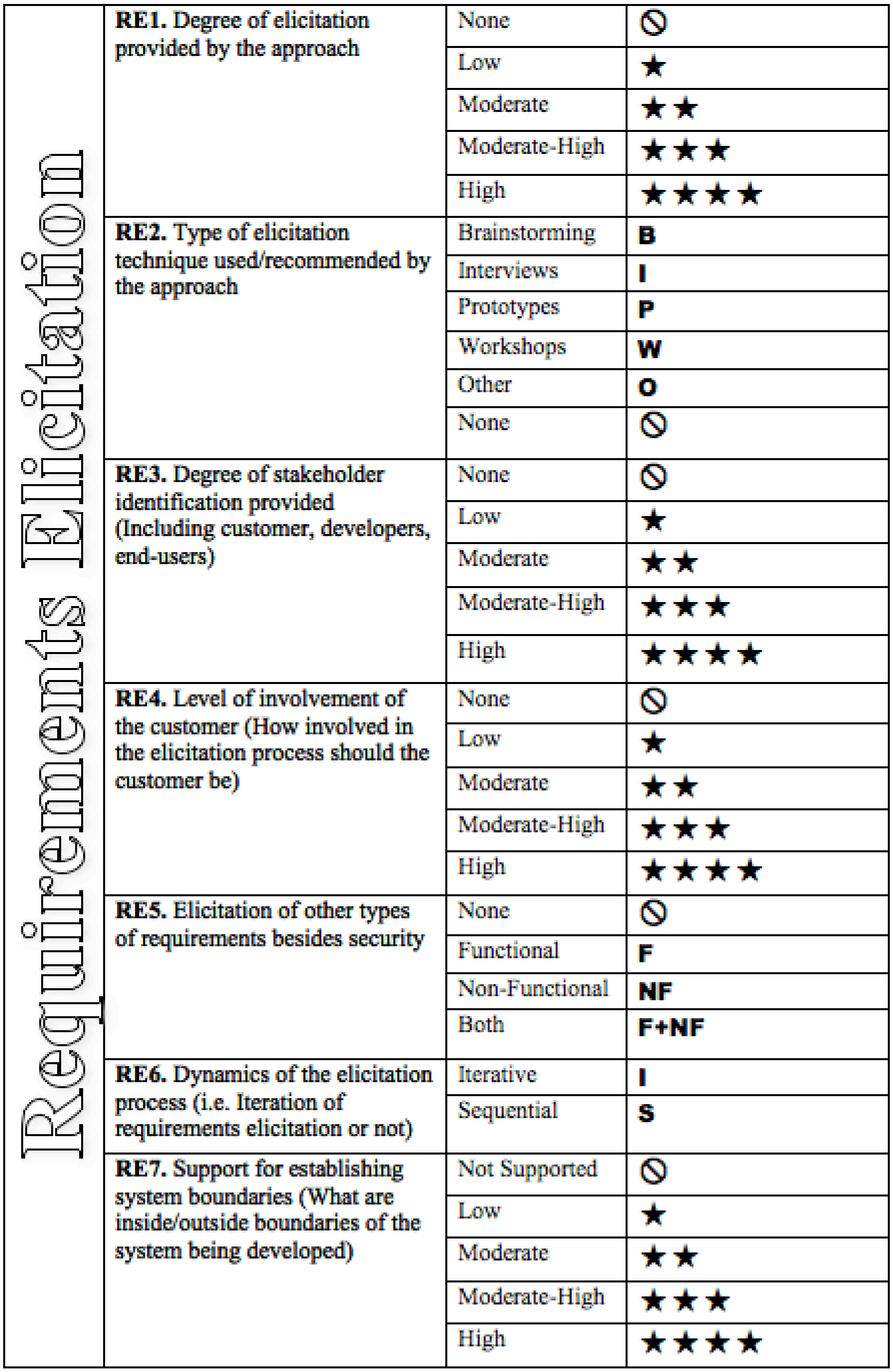
2. Gather the "wish list" for each relevant party.

3. Document and refine the "wish list" for each relevant party.

4. Integrate the wish lists across the various relevant parties, henceforth called viewpoints, thereby resolving the conflicts between the viewpoints.

5. Determine the nonfunctional requirements, such as performance and reliability issues, and state these in the requirements document.

For our survey, we wanted to understand not only if a certain approach supports security requirements elicitation, but also how well it does. Figure 3 shows the part of the framework specifically targeting the elicitation support that the approaches surveyed provide. We were mainly concerned with identifying important aspects about the elicitation process such as identifying the stakeholders, customer involvement, and the possibility to elicit other types of requirements besides security with the approach at hand. We have also identified how well each approach fulfills each question, not just if they support it or not.



Requirements Elicitation	**RE1.** Degree of elicitation provided by the approach	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RE2. Type of elicitation technique used/recommended by the approach	Brainstorming	**B**
		Interviews	**I**
		Prototypes	**P**
		Workshops	**W**
		Other	**O**
		None	⊘
	RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RE5. Elicitation of other types of requirements besides security	None	⊘
		Functional	**F**
		Non-Functional	**NF**
		Both	**F+NF**
	RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	Iterative	**I**
		Sequential	**S**
	RE7. Support for establishing system boundaries (What are inside/outside boundaries of the system being developed)	Not Supported	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★

Figure 3. Security Requirements Elicitation Framework

5.2.2 Security Requirements Analysis

The New York City Office of Technology tells us that the purpose of requirements analysis is to obtain a thorough and detailed understanding of the business need and to break it down into discrete requirements, which are then clearly defined, reviewed and agreed upon with the Customer Decision-Makers. Requirements Analysis provides the foundation for the desired product or services. These requirements will become the specifications if the procurement process is invoked [NY01]

Once the security requirements have been elicited, they need to be analyzed in order to ensure that their condition is a good starting point for specification; in the case that changes need to be made, we expect the approach to be able to tell us what these changes are and how they should be addressed. In our survey we probed this phase with a variety of questions including aspects such as completeness and clarity resolution; we are looking at each approach and determining if they help identify completeness and clarity issues as well as to help resolve them. We also consider important the ability of the analysis to help consider/identify alternative security requirements or additional ones that might have been missed during the elicitation phase. Figure 4 shows the complete set of questions aimed at security requirements analysis, along with their criteria.

Requirements Analysis	**RA1.** Type of overall analysis	None	⊘
		Internal (Ver.)	**I**
		External (Val.)	**E**
		Both	**I+E**
	RA2. Unambiguity resolution level of the analysis (Ambiguity issues can be detected and resolved through the analysis)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	Not Supported	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RA6. Analysis helps prevent security requirements conflict	Not Supported	⊘
		Marginally	★★
		Definitely	★★★

Figure 4. Security Requirements Analysis Framework

5.2.3 Security Requirements Specifications

While there exists a variety of definitions about what a requirements specifications is, we agree with the definition of Rombach [Rom90] who states that a specification is “a plan or standard that provides a description/characterization of a software product or process type.”

Often, a requirement specification is considered to be “good” based on a variety of characteristics. These characteristics often include correctness, unambiguity, completeness, consistency, verifiability, modifiability, and traceability. In our framework we have explored how good the specifications produced by the different approaches fulfill these characteristics.

In addition, IEEE [IEEE98] tells us that a good set of requirements specifications should fulfill the following,

1- Establish the basis for an agreement between the customers and the developers on what the software system is expected to do

2- Reduce the development effort

3- Provides a schedule for estimating costs and timelines

4- Provides a baseline for validation and verification

5- Serves as a basis for system enhancement

Figure 5 shows the framework of evaluation for the security requirements specification phase of the survey; as mentioned above, these questions and criteria have been formulated with the characteristics of a good specification in mind, as well as with the aspects described by the IEEE that a good specification should also fulfill.

Requirements Specification	**RS1.** The specification produced can be used as a baseline for system validation and/or verification once implemented	None	⊘
		Validation	**Va**
		Verification	**Ve**
		Both	**Va+Ve**
	RS2 The specification provides a basis for cost and/or time estimation of the overall development project	None	⊘
		Cost	**C**
		Time	**T**
		Both	**C+T**
	RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	Not Supported	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RS4. Consistency degree of the specification produced	Not Supported	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RS5. Support for specifying non-functional requirements other than security ones	Not Supported	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RS6. Overall clarity and understandability of the requirements specification	Not Supported	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RS7. Level of formality of the specification	Informal	**I**
		Semi-formal	**S**
		Formal	**F**
	RS8. Rigor of the specification process (how formal is the process itself)	Informal	**I**
		Semi-formal	**S**
		Formal	**F**
	RS9. Overall security level of the specification (How secure are the requirements that have just been specified?)	Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★

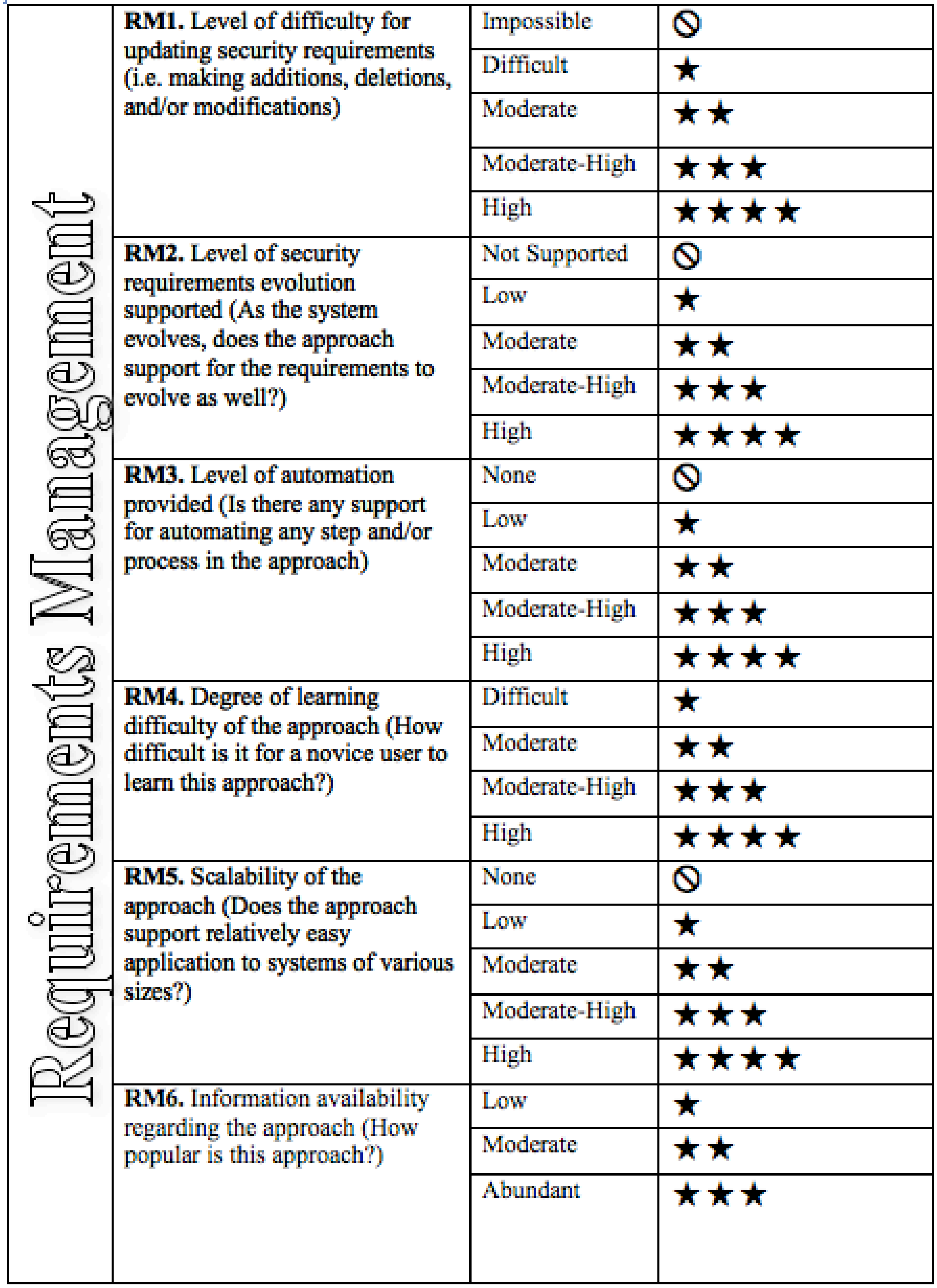
Figure 5. Security Requirements Specification Framework

5.2.4 Security Requirements Management

Requirements management is a very important phase in the engineering of security requirements; a recent survey of over 3800 European organizations in 17 countries found that most of the perceived software problems are in the area of requirements specification (>50%) and requirements management (50%) [ESI96].

Paulk et. al. asserts that "requirements management involves establishing and maintaining an agreement with the customer on the requirements for the software project" [Pau93]. For this survey we are interested in the "maintaining" part of the definition.

Once the security requirements have been elicited, analyzed, and specified, it is important for an approach to provide support that will enable one to manage them later on. Figure 6 shows the questions and evaluation criteria designed for the security requirements management phase of the survey. There are two main sets of questions; those that probe aspects about the support that the surveyed approaches provide for security requirements once they have been created, and questions that examine the approach for miscellaneous information useful during management. Some of the questions in the first set include the level of difficulty involved in updating the security requirements, does the approach provide any support for the evolution of the security requirements, and the level of automation provided by the approach. The second set includes questions like the learning difficulty of the approach as well as the amount of information available regarding it.



Requirements Management	**RM1.** Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	Impossible	⊘
		Difficult	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	Not Supported	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	Difficult	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	RM6. Information availability regarding the approach (How popular is this approach?)	Low	★
		Moderate	★★
		Abundant	★★★

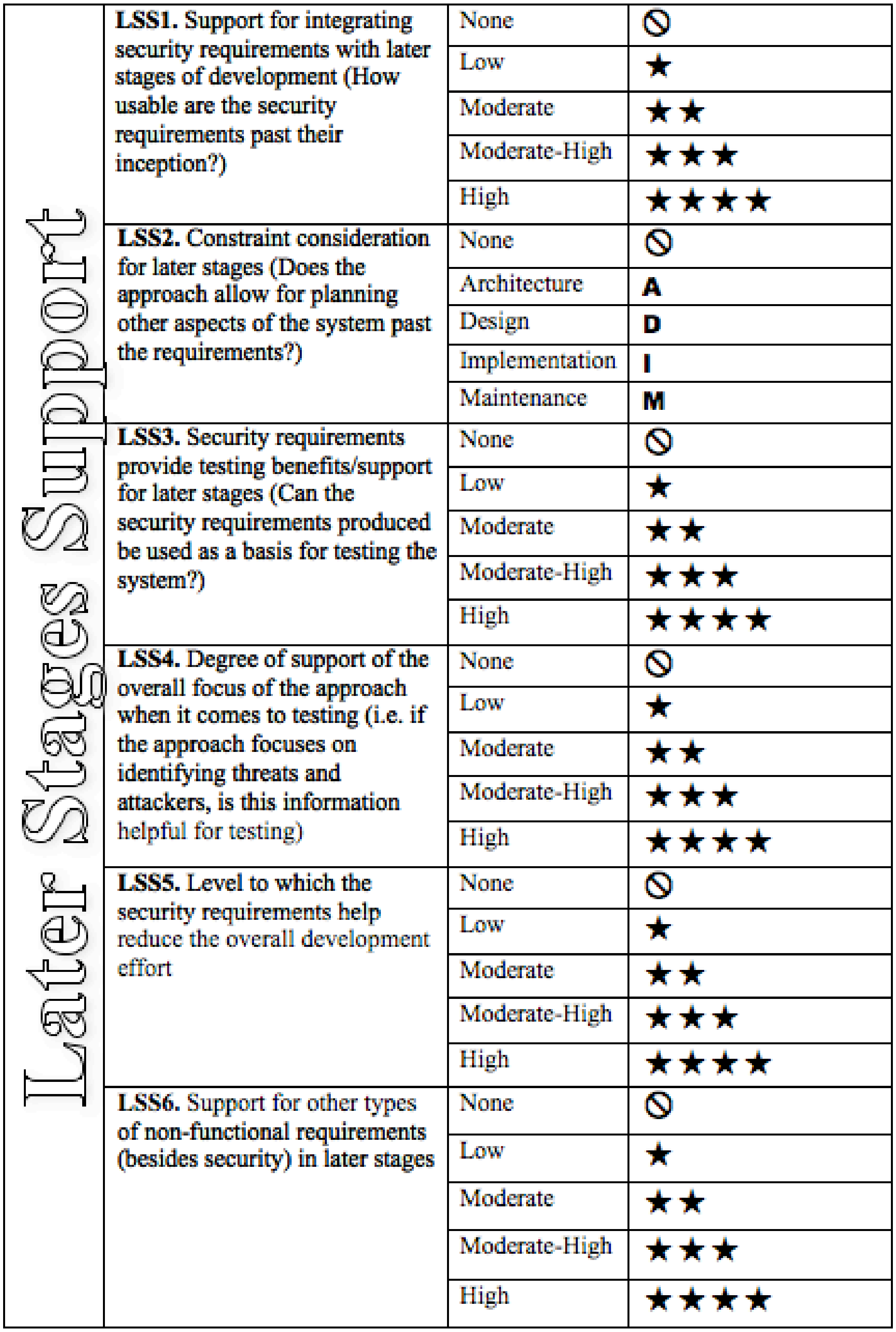
Figure 6. Security Requirements Management Framework

5.2.5 Later Stages Support for Security Requirements

As stated in the previous sections, we believe that it is important to not only be able to elicit, analyze, specify, and manage correctly security requirements, but it is also important to provide support for integrating them at later stages of the development cycle and ensuring their usefulness.

While we were not able to find much information during our survey about approaches that specifically support security requirements during later stages of development, we designed a set of questions and criteria for our framework to determine based on the information of each approach, how well they would provide this kind of support. The most important question that we ask in this part of the framework is if there is any support, either described specifically in the paper or that can be interpreted, for integrating the security requirements with later stages of development (i.e. architecture, design, implementation, testing, maintenance). Along this line we also probe about constraint consideration that the approach provides for any other stage; for example, some approaches might specifically support the specification of architectural constraints based on the security requirements. We are also interested in any testing benefits that the security requirements or the approach itself provides; this question should be looked at from the perspective of how useful during testing are the security requirements produced. We also ask how useful for testing is the focus of the approach; this refers more to the benefits of the process for testing, rather than the requirements themselves. Lastly, we inquire about support for other types of requirements, besides security ones, in later stages of development.

Figure 7 shows the framework for determining the level of later stages support provided by the surveyed methods.



Later Stages Support	**LSS1.** Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	None	⊘
		Architecture	A
		Design	D
		Implementation	I
		Maintenance	M
	LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	LSS4. Degree of support of the overall focus of the approach when it comes to testing (i.e. if the approach focuses on identifying threats and attackers, is this information helpful for testing)	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	LSS5. Level to which the security requirements help reduce the overall development effort	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★
	LSS6. Support for other types of non-functional requirements (besides security) in later stages	None	⊘
		Low	★
		Moderate	★★
		Moderate-High	★★★
		High	★★★★

Figure 7. Later Stages Support for Security Requirements Framework

5.3 Evaluation Framework Limitations

There are some limitations to our framework that need to be discussed. The most significant limitation of our framework is its inherent subjectivity; 30 of the 34 questions of the framework have been subjectively answered. Even though the rating (number of stars) that each answer has been given has been done with as much information as possible, it is nonetheless subjective. This is an expected limitation of our framework, but believe that the explanations associated with each rated answer will help explain the reason behind the decision.

The second limitation our framework suffers from has to do with the value of each star. While we realize that stars might be more valuable (harder to obtain) at specific questions, they are all worth the same. For example, there could only be one approach that obtains a star for a specific question, while the rest obtain 0 stars for the same question; in this case, this star is "worth" a lot more since an approach must have tremendous support to obtain a star for it. In order to help reduce the subjectivety of the survey wherever possible and to help "standarize" the star counts, we decided to make each star worth the same regardless of questions or phases.

Our last limitation has to do with the availability of information for each approach surveyed. While we explain in the following section our criteria for selecting the surveyed approaches, there was more information available for some than others. This means that while the rating of the answers have been done as informed as possible, some answers might be more accurate than others based on the amount of information regarding each approach surveyed.

6 Surveyed Approaches

For the purposes of this survey, we focus only on those approaches that proactively address the issue of security. We came across a variety of approaches that could be adapted to engineer security requirements, but we did not consider them because they make no mention of security as they currently stand. Over 30 SRE were originally considered,

1. Knowledge Agent-oriented System (KAOS)
2. Risk Analysis
3. Security Patterns
4. Security Design Analysis (SeDaN)
5. Abuse Cases
6. Software Cost Reduction
7. Threat Trees
8. Fault Trees
9. Problem Frames
10. Security Use Cases
11. Simple Reuse of Software Requirements (SIREN)
12. Threat Modeling for Security Requirements
13. Agile Security Requirements Engineering

14. Security Models
15. Security Development Lifecycle Tool (SDL)
16. Controlled Requirements Expression (CORE)
17. Joint Application Development (JAD)
18. Issue-based information systems (IBIS)
19. Critical discourse analysis (CDA)
20. Accelerated Requirements Method (ARM)
21. Quality Function Deployment (QFD)
22. Misuse Cases
23. Abuser Stories
24. Secure TROPOS
25. Security Problem Frames
26. Anti-models
27. i* Security Requirements
28. Common Criteria
29. System Quality Requirements Engineering (SQUARE)
30. Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE)
31. Attack Trees
32. Usage-centric Security Requirements Engineering (USeR)
33. Comprehensive Lightweight Application Security Process (CLASP)

The number of SRE approaches surveyed in detail was later reduced to 12; this was necessary in order to keep the literature survey within a manageable size. These 12 approaches were selected based on two main factors,

- Maturity. The first factor is their "seniority" in the field, meaning that we strived for surveying those approaches that are most popular.
- Information availability. The second factor has to do with the amount of information available about each approach; we decided to stay away from those approaches that were nothing more than a "position" paper or "future directions" paper, we wanted to explore those that were more mature when it came to their development.

Once the 12 approaches were selected, we divided them into two main groups; the first group is composed of those approaches that in one way or another have been adapted from exisitng approaches (that do not support security) to address security. The second group are approaches that have been developed specifically with security at heart; these approaches have not been adapted from exisiting ones, but have rather built security into the approach from the "ground-up." Figure 8 shows the approaches surveyed and the category they fall into (derived or original); for the ones that have been derived from existing ones we show the parent approach. We have six approaches for each category.

Approaches Adapted from Existing Ones to Address Security (Derived)	**Approaches Developed Specifically for Security (Original)**
Use Cases 1. Misuse Cases	7. Common Criteria
User Stories 2. Abuser Stories	8. SQUARE
TROPOS 3. Secure TROPOS	9. OCTAVE
Problem Frames 4. Security Problem Frames	10. Attack Trees
Obstacle Analysis 5. Anti-models	11. USeR Method
i* 6. i* Agent-based Requirements for Security	12. CLASP

Figure 8. Surveyed Approaches

6.1 Misuse Cases

Misuse cases [Hop04, Ale03] have been derived from use cases to look at the system from the point of view of a malicious user. Use cases have become popular for eliciting, communicating and documenting various types of requirements [Rum94, Kul00, Wei98]. Use cases are great in the fact that they provide a different approach to requirements gathering, a more visual one. As it turns out, many groups of stakeholders are more comfortable with descriptions of operational activity paths than with declarative specifications of software requirements. Misuse cases allow you to harness this advantages, but in a different way.

According to Jacobson et al [Jac92], a misuse case is a use case from the point of view of an actor hostile to the system under design. Its goal is not a system function but a threat posed by that hostile actor. Some misuse cases occur in highly specific situations, whereas others continually threaten systems. For instance, a car is most likely to be stolen when parked and unattended, whereas a web server might suffer a denial-of-service attack at any time.

6.1.1 Framework Application

Table 1 shows the results of applying our framework to misuse cases.

Misuse Cases

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★★★★	Misuse cases **highly** help elicit security requirements
RE2. Type of elicitation technique used/recommended by the approach	**B**	The elicitation technique mainly used is **brainstorming** sessions where both the customers and developers interact
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★	There is a **moderate** degree of identification of stakeholders. Misuse cases help delineate the relationship that developers and customers will have. They also help in identifying possible attackers
RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	★★★	Misuse cases rely a lot on the customers, and therefore they are **moderately-highly** involved in the elicitation process. Additionally, the success of misuse cases also depends on the quality of communication between the developers and customers
RE5. Elicitation of other types of	**NF**	Other **non-functional requirements**

requirements besides security	★★★	like **safety** can be elicited with misuse cases (**moderate-high** support)
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	**I**	Security requirements are elicited **iteratively**; meaning that developers go back to the customers constantly to help refine the security requirements elicited so far
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★★	**Moderate.**
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	**I** ★★★ **+** **E** ★	Misuse cases provide **both**, **internal and external** analysis of the security requirements elicited. The internal analysis (verification) is inherited from use cases and is **moderate-high** the external is **low**
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★	**Low.** While the customers are highly involved in helping iteratively shape the security requirements, the inherent informality of misuse cases still leaves plenty of room for ambiguity to slip through the analysis phase.
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★★	There is **moderate** support for ensuring completeness.
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★	Their informality once again plays an important role in how clear the misuse cases are. They provide a **low** clarity resolution level; while the customer can help in clarifying them, there is too much risked with the natural language used
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★★	**Moderate** reasoning about alternative security requirements is provided by misuse cases. The approach helps you consider numerous issues that could cause the system to fail; chances are that a number of these issues have not been considered during the elicitation process.

RA6. Analysis helps prevent security requirements conflict	★★★	Misuse cases **definitely** help you resolve conflicts among security requirements. Misuse cases have additional relationships referred to as "aggravates" and "conflicts with" for this very purpose.
REQUIREMENTS SPECIFICATION		
RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	Va ★★ + Ve ★★	The specification produced could support **both** the **validation and verification** of the resulting system both **moderately**
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	C ★★	Specifications produced with misuse cases can help with tradeoff analysis involving **cost** (**moderate** support)
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★	**Low**. Misuse cases on their own do not really provide much support, nor emphasize traceability as part of the specifications
RS4. Consistency degree of the specification produced	★★	The specifications produced are **moderately** consistent. Further support is offered through tools designed for use cases (which could be applied to misuse cases as well) that check for consistency of the security requirements
RS5. Support for specifying non-functional requirements other than security ones	★★★	**Moderate-high.** We believe that misuse cases are easily adaptable to cover other ranges of non-functional requirements like safety and privacy for example.
RS6. Overall clarity and understandability of the requirements specification	★★	**Moderate** overall clarity and understandability due to their informal nature
RS7. Level of formality of the specification	I	The specifications produced are **informal**
RS8. Rigor of the specification process (how formal is the process itself)	I	The approach is very **informal**

RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★	**Moderate.** While the specifications produced are secure, they are not as secure as they could be due to informal nature of misuse cases
REQUIREMENTS MANAGEMENT		
RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★★★★	The simplicity and informality of misuse cases makes it very **easy** to update when needed
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	⃠	**Not Supported.**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	★	**Low.** Most misuse cases must be developed manually, this because misuse cases identify vulnerabilities which is difficult to do automatically (unless the systems are so similar that the same vulnerabilities could apply to both)
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★★★★	Misuse cases are **easy** to learn; a benefit derived directly from use cases
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★	**Low**. The approach works well with relatively small systems; when the systems become larger it is difficult to systematically develop security requirements using misuse cases
RM6. Information availability regarding the approach (How popular is this approach?)	★★★	There is **abundant** information available.
LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	⃠	While there is **no** explicit information about any support for integrating misuse cases with later stages of development, there is some support for integrating use cases; this knowledge could potentially be applied to misuse cases

LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	⃠	**None.**
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★★	**Moderate.** Misuse cases can help you identify possible failure modes as well as exceptions that the system can incur, these can be modeled as test cases in order to test boundary conditions
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★★★	The main focus of misuse cases is to determine ways that the system should not be used from the perspective of the possible attacker; this focus **moderately-highly** supports testing efforts
LSS5. Level to which the security requirements help reduce the overall development effort	★★	Misuse cases can **moderately** help reduce the development effort. Misuse cases help in considering solutions that mitigate the misuse cases themselves; this information can be applied during development to narrow down security strategies.
LSS6. Support for other types of non-functional requirements (besides security) in later stages	★	**Low**. Misuse cases could provide some sort of support for safety requirements

Table 1. Misuse Cases Framework Application

6.2 Abuser Stories

Abuser stories [Pet07, Pet05] have been adapted from user stories to address security. An Abuser story is a textual description of the malicious interaction between a threat agent and the system itself that, if successful, results in the increase of risk to the assets valued by the customer. An example of a simplified abuser story is: *"A participant could modify the proposal of a competitor to make it look bad."*

Abuser stories are discussed with the customer to ensure their relevance and importance. Finding good abuser stories is a mostly brainstorming activity. However, using resources such as attack patterns can be helpful here [Hog04]. Abuser stories can also be used as a starting point to security testing of the system. Abuser stories have been proposed to help engineer security requirements for XP projects. [Bos061]

6.2.1 Framework Application

Table 2 shows the results of applying our framework to abuser stories.

Abuser Stories

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★★★	**Moderate-High**. Abuser stories help elicit security requirements
RE2. Type of elicitation technique used/recommended by the approach	I	The main technique used is **interviews**
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★★★	There is a **high** degree of identification of stakeholders. It helps identify not only the key customers and developers involved, but also helps envision the relationship between them.
RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	★★★	Customers are **moderately-highly** involved; they check the abuser stories constantly for accuracy and aid in updating them to accommodate changes
RE5. Elicitation of other types of requirements besides security	⃠	**Not Supported**
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	I	The elicitation process is **iterative.** Developers bring to the customers the requirements elicited to help them ensure they are correct

RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★	**Low.** This is not explicitly supported by the approach, but due to the high customer involvement, elicitation for scope related abuser stories could be derived
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	I ★	The approach mainly provides an **internal analysis**, but it there is **low** support for it
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★	There is **low** support for resolving ambiguities in the requirements
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★	There is **low** support for ensuring that the requirements that have been elicited are complete
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★	There is **moderate** support for ensuring that the abuser stories are clear; this because the customer checks them constantly and in turn helps refine them
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	🚫	**Not Supported**
RA6. Analysis helps prevent security requirements conflict	🚫	**Not Supported.** There is no mention about the approach providing any kind of support for resolving conflicts among requirements.
REQUIREMENTS SPECIFICATION		
RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	Va ★	User stories have shown potential for helping **validate** a system; we think that abuser stories could also serve that purpose. Based on user stories' background, there could be **low** support for using the security specifications as a validation point
RS2. The specification provides a basis for cost and/or time	C ★★	Abuser stories can help in estimating **both time and cost**. Abuser stories

estimation of the overall development project	+ T ★★★	provide brief estimations on how long each story should take to implement; **moderate-high** support. Additionally, they also help optimize the net cost as well as bringing into account additional costs of possible attacks; thus providing **moderate** support.
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★★★★	Abuser stories can prove to produce **highly** traceable security requirements; this is one of their highlights and aid to the agile requirements process
RS4. Consistency degree of the specification produced	★	Besides the possible consistency benefits obtained from a high customer involvement, abuser stories produce **low** consistency in security requirements; mainly due to their informal nature
RS5. Support for specifying non-functional requirements other than security ones	⃠	**Not Supported**
RS6. Overall clarity and understandability of the requirements specification	★★	**Moderate**. While abuser stories are easy to understand due to their simplicity, they can be very unclear due to their informality
RS7. Level of formality of the specification	I	**Informal** approach.
RS8. Rigor of the specification process (how formal is the process itself)	I	**Informal** approach.
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★	**Low**
REQUIREMENTS MANAGEMENT		
RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★★★★	Their simplicity makes updating abuser stories **easy**; as there is not much effect on other abuser stories when one is updated
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	⃠	**Not Supported**

RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	⊘	**None**
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★★★	This approach is **moderate-easy** to learn as it does not involve any complex syntax/semantics, nor an outrageous number of steps
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★	**Low**. Because they are so simple, it seems difficult to be able to develop security requirements for a large project based on them
RM6. Information availability regarding the approach (How popular is this approach?)	★	**Low**.
LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	⊘	**None.**
LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	I ★★	**Implementation** constraints could be derived based on the information regarding the time needed to implement each story; **moderate** level of support for implementation
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★★	There is **moderate** support for testing that is made explicit by abuser stories
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★★	The main focus of the approach, abuser stories, provide **moderate** support for testing due to the fact that each story could easily be implemented as a test case (or series of related test cases) in order to test specific aspects identified in the security requirements

LSS5. Level to which the security requirements help reduce the overall development effort	★★	Provides a **moderate** level of support for reducing the effort in developing the system.
LSS6. Support for other types of non-functional requirements (besides security) in later stages	🛇	**None**.

Table 2. Abuser Stories Framework Application

6.3 Secure Tropos

Secure Tropos is an extension of the Tropos methodology [Mou06]. Tropos [Bre04] is an agent oriented software engineering methodology, in which notions such as actors, goals, soft-goals, tasks, resources, and intentional dependencies are used in all the phases of the system development from the first phases of the early analysis, down to the actual implementation. The Tropos methodology is mainly based on four phases [1]: early requirements analysis, late requirements analysis, architectural design, and the detailed design phase.

Although, the Tropos methodology was not designed with security in mind, there is a set of security related concepts available [Mou03, MoG03, MGS03] that enable it to model security issues. This security-oriented extension, known as Secure Tropos, is built upon a variety of concepts, including security constraints, secure dependencies, and secure goals. A security constraint is a restriction related to security issues, which can influence the analysis and design of the information system under development. Secure dependencies introduce security constraint(s) that must be fulfilled for the dependency to be satisfied. A secure goal represents the strategic interests of an actor with respect to security. Secure goals are mainly introduced in order to achieve possible security constraints that are imposed to an actor or exist in the system [Gio06].

6.3.1 Framework Application

Table 3 shows the results of applying our framework to Secure Tropos.

Secure TROPOS

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★	Secure TROPOS provides **low** support for eliciting security requirements
RE2. Type of elicitation technique used/recommended by the approach	**Ot**	**Other.** Security requirements are elicited using the concept of constraints.
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★	**Moderate.** The approach helps identify besides customers and developers, relevant actors in the system-to be, along with their respective dependencies
RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	★	**Low.**

RE5. Elicitation of other types of requirements besides security	**NF** ★	This approach can help elicit other **non-functional requirement** like privacy and usability (low support)
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	**I**	**Iterative**. The requirements are elicited through an incremental refinement process
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★	**Low**.
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	**E** ★★	**Moderate external** analysis of the security requirements elicited is provided. Security constraints are imposed on the stakeholders to help validate the security requirements
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★	**Low.**
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★★	**Moderate.** Additionally, the variety of models that are provided can help to systematically determine how complete the security requirements are
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★	**Moderate.** Secure TROPOS analyzes in depth the goals of each actor in the security requirements and the security constraints on them to help make them as clear as possible
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	⊘	**None.**
RA6. Analysis helps prevent security requirements conflict	★★	**Marginally**. Dependencies between the actors in the security requirements are modeled; this knowledge can prove useful in resolving conflicts

REQUIREMENTS SPECIFICATION		
RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	**Va** ★★★★	Secure TROPOS specifications can **highly** help with two kinds of **validations**; model and design validation
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	**C** ★ + **T** ★	**Cost** and **time** estimations are **both** possible with Secure TROPOS. The approach strives for cost-effective protection as well as security reference modeling aims specifically at helping save time (**low** support for both)
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★	Traceability of the specification is **low**
RS4. Consistency degree of the specification produced	★★★	A **moderate-high** level of consistency is provided. There is support for not only ensuring consistency of the specification produced but also of the models developed with it through the use of the outer-model rules provided
RS5. Support for specifying non-functional requirements other than security ones	★★	**Moderate.** TROPOS allows for the specification of other non-functional requirements as "soft-goals;" this could also be applied to Secure TROPOS in order to support other non-functional requirements
RS6. Overall clarity and understandability of the requirements specification	★★	**Moderate**.
RS7. Level of formality of the specification	**S**	The specifications are **semi-formal**
RS8. Rigor of the specification process (how formal is the process itself)	**F**	The approach is **formal**
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★	**Moderate.**

REQUIREMENTS MANAGEMENT		
RM1. Level of difficulty for updating security requirements	★★	**Moderate.** The approach provides tool support for checking the consistency of

(i.e. making additions, deletions, and/or modifications)		requirements; this tool can be applied after updates are done to identify effects on other security requirements
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	⃠	**Not Supported.**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	★★★	**Moderate-high.** The approach provides tool support for automatically checking the consistency of security requirements. Additionally, it supports the idea of "autonomy reasoning" through actor and goal diagrams
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★★	Learning is **moderately** difficult. The syntax and semantics used seem tricky to learn
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★★★	Scalability is **high**. The approach is easily extensible to accommodate for larger systems
RM6. Information availability regarding the approach (How popular is this approach?)	★★	**Moderate.**
LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	★★	**Moderate.** There is support for helping transform the specification into a design
LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	A ★★ + D ★★	Secure TROPOS helps determine what needs to be addressed at **design** time and **architecture** using an actor diagram (both **moderately**)

LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★★	**Moderate**. Can potentially help in testing our design against the initial security requirements
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★	**Low.**
LSS5. Level to which the security requirements help reduce the overall development effort	★	**Low.**
LSS6. Support for other types of non-functional requirements (besides security) in later stages	★★★	**Moderate-high**. Privacy, availability, and integrity requirements could also be supported at design

Table 3. Abuser Stories Framework Application

6.4 Security Problem Frames

Security problem frames [Hat07, Hat05, HaH07, Sec1] are an adaptation from the existing problem frames approach. The basic idea of problem frames is to make use of special patterns defined for structuring, characterizing, and analyzing problems that occur frequently in requirements engineering by Michael Jackson [Jac01]. The advantage of using problem frames in requirements engineering is that problems are mapped to well-known problem classes that are practically relevant. Once a problem is successfully fitted to a problem frame, its most important characteristics are known, because these are shared by all problems fitting that frame.

Security problem frames [Hat06] are special kinds of problem frames, which consider security requirements. They strictly refer to the problems concerning security, without anticipating solutions. For example, we may require that data is kept confidential during transmission without being obliged to mention encryption, which is a means to achieve confidentiality. As in problem frames, which are targeted at functional requirements, the same process can be used to structure, characterize, and analyze security-specific problems.

6.4.1 Framework Application

Table 4 shows the results of applying our framework to security problem frames.

Security Problem Frames

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	🚫	Security problem frames counts on requirements being **already elicited** with another approach (like CREE for example). Their main support for security requirements starts at the analysis level
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	E ★★	The approach mainly provides a **moderate external analysis**
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★★★	There is **moderate-high** support for resolving ambiguities. “Concretized security frames” help in ensuring that the security requirements are more specific; this specificity can help in coping with ambiguity
RA3. Completeness resolution level of the analysis (Analysis	★★★★	There is **high** support for ensuring that the requirements that have been elicited are

can help determine if the security requirements are complete)		complete. Security requirements are analyzed using an iterative process, in which “sub-problems” are created until all the preconditions of other security problem frames can either be proven to be true or assumed to be true. This iterative process helps guarantee that the set of security requirements are complete
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★	**Moderate.** The continuous decomposition of problem frames helps in making them clearer
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★★★	**Moderate-high** support achieved mainly by the fact that security problem frames provides developers with a “related section” of each frame used so far in order to see other security requirements that should be associated with the current frame
RA6. Analysis helps prevent security requirements conflict	★★	**Marginally.** Requirements conflict could be avoided by relying on proven problem frame dependencies and structures
REQUIREMENTS SPECIFICATION		
RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	⊘	**Not Supported.**
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	**T** ★★★	The specification could potentially provide a **moderate-high** basis for **time** estimation based on the time it took to implement pre-existing frames that the current security problem frames could be based on
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★	**Low**. There is no evidence that points to the fact that the specifications produced by security problem frames could be traceable; nor they mention any type of support for achieving this; but we believe that the knowledge about their dependencies could be exploited to support traceability
RS4. Consistency degree of the specification produced	★★	Due to explicit knowledge of dependencies among security problem frames, this helps make them **moderately** consistent

RS5. Support for specifying non-functional requirements other than security ones	★	**Low.** Even though problem frames could potentially be used to specify other non-functional requirements like confidentiality, there is not much support described
RS6. Overall clarity and understandability of the requirements specification	★	**Low.**
RS7. Level of formality of the specification	**F**	The specifications produced are **formal** as they have their specific syntax and structure
RS8. Rigor of the specification process (how formal is the process itself)	**S**	While the approach seems to be **semi-formal**, certain aspects of it could potentially be formalized using Object Z
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★	**Moderate**
REQUIREMENTS MANAGEMENT		
RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★★	Updating security problem frames could be **moderately** difficult, because while there are existing patterns that the updated frame could fit into, you still have to consider which one it is as well as how it would affect its dependencies
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	★	**Low.** The knowledge of how concretized security problem frames evolve from normal security problem frames could be applied to their evolution in general; but this is just a speculation
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	⃠	**None**
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★★	**Moderate**, as you need background knowledge on problem frames in general as well as existing methods for elicitation of security requirements

RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★★	**Moderate-high**. The more extensive the library of patterns that you have, the easier it would be to apply the approach to a much larger system
RM6. Information availability regarding the approach (How popular is this approach?)	★★	There is a **moderate** amount of information available regarding security problem frames.
LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	⃠	**None.** There is no explicit support/guidance for making the security requirements usable later on
LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	A ★★★★	There is **high** support for associating security problem frames with defined **architectural** patterns; information that can be used in order to establish architectural (and possibly design) constraints based on the security requirements
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	⃠	**None.** There is no explicit consideration of the testing phase by security problem frames
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★	While there is no explicit sense of security benefits from problem frames, its focus could aid at a **low** level
LSS5. Level to which the security requirements help reduce the overall development effort	★★★	**Moderate-high.** A benefit of having a complete set of security requirements is that through the support of concretized problem frames, you also have a complete set of solution approaches to each security problem frame. This knowledge can be used in order to develop specific security mitigation mechanisms that embody each one of the solution approaches.

LSS6. Support for other types of non-functional requirements (besides security) in later stages	⊘	**None**.

Table 4. Security Problem Frames Framework Application

6.5 Anti-Models

Anti-models [Hal04, Sin03] have been derived from existing work in obstacle analysis [Lut07]. In anti-models, the basic notion is to develop two kinds of models simultaneously. The first is a model of the system-to-be that covers both the software and its environment and inter-relates their goals, agents, objects, operations, requirements and assumptions. The second model is the actual anti-model, which is derived from the first model as it is being developed. This anti-model exhibits how specifications of model elements could be maliciously threatened, why and by whom.

In this type of development, security requirements are elaborated systematically by iterating the following steps:
1- instantiation of specification patterns associated with property classes such as confidentiality, privacy, integrity, availability, authentication or non-repudiation,
2- Develop anti-model specifications threatening such specifications,
3- Generate alternative countermeasures to such threats and define new requirements by selection of alternatives that best meet other quality requirements from the model. [Lam04]

6.5.1 Framework Application

Table 5 shows the results of applying our framework to the anti-models approach.

Anti-Models

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★★	Anti-models can help in eliciting security requirements **moderately**, as it elicits them from the perspective of eliciting security-related goals from instantiations of specification patterns
RE2. Type of elicitation technique used/recommended by the approach	**Ot**	**Other**. The main technique used is the negation of goals that have been obtained from the customer
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★★	There is a **moderate-high** degree of identification of stakeholders. Besides customers and developers that need to be involved, it also helps identify possible attackers
RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	★★	**Moderate.** In order to obtain answers to key aspects of the security requirements like “who can benefit from this anti-goal?” customers are involved.
RE4. Level of involvement of the customer (How involved in	★★	**Moderate.** In order to obtain answers to key aspects of the security requirements

RE5. Elicitation of other types of requirements besides security	**NF** ★★★	Other **non-functional requirements** like **safety** ones could also be elicited with anti-models; these requirements could be elicited **moderately-highly** with anti-models
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	**I**	Security requirements are elicited through an **iteration** of three major steps
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★★★	**Moderate-high.** The security requirements elicited with anti-models can help developers determine different characteristics of the different problem domains at hand, along with their interactions. This information can be used to determine what is outside of the scope of the system

REQUIREMENTS ANALYSIS

RA1. Type of overall analysis	**I** ★	There is **low** support to **internally** analyze the security requirements elicited
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★★★	**Moderate-high**. The iteration of the anti-goals that have already being defined helps detect possible ambiguity issues, and helps resolve them.
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★	There is **low** support for ensuring completeness. The approach argues strongly for the importance of complete requirements, but there is no explicit step/tool for checking completeness.
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★★	The continuous iteration of the anti-goals provides **moderate-high** support for ensuring that the security requirements elicited are clear
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★★★	**Moderate-high.** The approach helps to consider alternative countermeasures to the anti-models; this knowledge can be used to determine if additional and/or alternative security requirements are necessary.
RA6. Analysis helps prevent security requirements conflict	★★	**Marginal** support for dealing with conflicts in the requirements.
RA6. Analysis helps prevent security requirements conflict	★★	**Marginal** support for dealing with conflicts in the requirements.

REQUIREMENTS SPECIFICATION

RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	Ve ★★	The specification could potentially be used to help **verify** the implemented system; this support would be at the most of **moderate** benefit to the overall verification process
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	⃠	**None.**
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★★	The specifications produced seem to be **marginally** traceable; this because there is no explicit call to make them as traceable as possible, but there is nonetheless techniques available to provide traceability management
RS4. Consistency degree of the specification produced	★★	Due to explicit knowledge of dependencies among security problem frames, this helps make them **moderately** consistent
RS5. Support for specifying non-functional requirements other than security ones	★★★★	**High.** There is support for also specifying privacy and integrity requirements.
RS6. Overall clarity and understandability of the requirements specification	★	**Low.**
RS7. Level of formality of the specification	S	**Semi-formal.**
RS8. Rigor of the specification process (how formal is the process itself)	F	The process is **formal** because among other things it uses first order temporal logic as well as and/or goal refinement; the approach also provides a syntax and semantics for the specification
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★	**Moderate.** The approach also provides options for different levels of security assurance in the specifications

REQUIREMENTS MANAGEMENT

RM1. Level of difficulty for	★	**Difficult**. The formality involved in the
RM1. Level of difficulty for updating security requirements	★	**Difficult**. The formality involved in the process of developing anti-models (i.e

RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	⃠	**Not Supported.**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	★★	**Moderate.** While there are no explicit automation measures provided, there is a real-time temporal logic that is offered by the developers of the approach to help formalize anti-goals that could then serve as the input for other tools that can help in generating anti-model scenarios automatically
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★	**Difficult**, as you need background knowledge on various aspects like goals and goal refinement techniques in order to make the approach as effective as possible
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★★★	**High**. We believe that the approach can be applied to systems of various sizes; there is support added to enable an incremental approach to developing security requirements using anti-models
RM6. Information availability regarding the approach (How popular is this approach?)	★	There is a **limited** amount of information available.
LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	⃠	**None.**
LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	⃠	**None.**
LSS3. Security requirements provide testing benefits/support for later	★	**Low.** There is the possibility that some of the alternative countermeasures that the approach helps you consider could become

LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★	**Low.** There is the possibility that some of the alternative countermeasures that the approach helps you consider could become possible test cases; but there is no concrete information regarding this
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★	The overall focus of anti-models can provide a **low** level of support for testing
LSS5. Level to which the security requirements help reduce the overall development effort	★★	**Moderate** level of support for reducing the efforts of developing the system. The countermeasures identified can help anticipate a variety of aspects when it comes to developing specific security mechanisms

Table 5. Anti-Models Framework Application

6.6 i* Agent-based Requirements for Security

The i* security requirements approach is derived from the i* methodology [Luc04]. The i* methodology is an agent-oriented requirements modeling language. The i* agent-based requirements for security approach [Liu03] is a methodological framework for analyzing security requirements based on the concept of strategic social actors. The strength of the framework is that it offers a set analysis techniques aimed at helping stakeholders involved in the development of the system with a variety of issues. These issues include better understanding the threats and vulnerabilities involved, the possible countermeasures, and how to combine them to achieve the desired security level.

The analysis techniques provided include,
- Attcker Analysis. Identifies potential system abusers and their malicious intents.
- Dependency Vulnerability Analysis. Identifies the vulnerable points in the dependency network
- Countermeasure Analysis. Decisions are made on how to protect security from the potential attackers and vulnerabilities
- Access Control Analysis. Uses i* models to refine a possible countermeasure solution and bring it closer to a design state

6.6.1 Framework Application

Table 6 shows the results of applying our framework to the i* security requirements approach.

i Agent-Based Security Requirements*

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★★	i* for security requirements can help you **moderately** elicit security requirements due to the fact that it does this mainly from decomposing and analyzing existing documentation
RE2. Type of elicitation technique used/recommended by the approach	I + Ot	The main technique used is **interviews** as well as existing documentation analysis
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★★★	There is a **high** degree of identification of stakeholders, ranging from customers to actors and agents.
RE4. Level of involvement of the customer (How involved in the elicitation process should the	★★★	Customers are **moderately-highly** involved, not only to help elicit requirements but also to help verify them

customer be)		
RE5. Elicitation of other types of requirements besides security	**NF** ★	This approach can also help you elicit other **non-functional** requirements, like privacy (but there is **low** support for this)
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	**I**	The elicitation process is **iterative**
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★	**Low**. The scheme provided can help in determining what is outside of the scope of the system

REQUIREMENTS ANALYSIS

RA1. Type of overall analysis	**I** ★★	Model checking can be used in order to provide a **moderate** level of **internal analysis**
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★	There is **low** support for identifying ambiguities in the security requirements elicited and resolving them
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★★	The analysis **moderately** determines that the security requirements elicited are complete
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★★	The high involvement of the customer is crucial in enabling the analysis to **moderately-highly** clarify the security requirements
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★★	**Moderate** support is provided through the inherent nature of agent-based models in order to help identify alternative security requirements
RA6. Analysis helps prevent security requirements conflict	★★★	**Definitely** the richer description and analysis techniques that this approach supports help in detecting and resolving possible conflicts in the requirements

REQUIREMENTS SPECIFICATION		
RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	Va ★	There is not much information regarding this aspect, but based on i* requirements in general, the specification could potentially be used to provide a **low** degree of **validation**
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	C ★★	**Cost**. The approach helps in analyzing possible tradeoffs that can occur between security and cost when it comes to implementing the system (**moderate** support)
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★★	A **moderate** level of traceability can be achieved in the specification through the use of i* dependency modeling.
RS4. Consistency degree of the specification produced	★★★★	**Highly** consistent specifications can be produced if Telos (a tool provided by i*) is applied during the process; Telos helps check the consistency between the security requirements and the models
RS5. Support for specifying non-functional requirements other than security ones	★★★★	The approach provides a **high** degree of support for also specifying privacy requirements
RS6. Overall clarity and understandability of the requirements specification	★★	**Moderate**.
RS7. Level of formality of the specification	S	**Semi-formal** specifications produced.
RS8. Rigor of the specification process (how formal is the process itself)	F	**Formal** approach.
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★★★	**Highly** secure specifications can be produced with this approach; mainly due to the fact that it helps perform additional security-related activities like vulnerability analysis, attacker identification, and countermeasure identification. In addition, it also has a rule of considering all of the actors of the system as “guilty until proven innocent.”

REQUIREMENTS MANAGEMENT		
RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★	Since the security requirements can prove to be highly dependent on one another, updating one can prove to be **difficult**
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	⃠	**Not Supported**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	⃠	**None**
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★★	**Moderate**
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★	If Telos is also applied to the models, it can provide a **moderate** level of scalability for larger project
RM6. Information availability regarding the approach (How popular is this approach?)	★	**Low**. Information regarding this approach is very rare.
LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	★	**Low**. There exists the possibility of assistance in the transition into design through the concept of access control analysis that they describe
LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	D ★★ + I	Provides constraint **moderate** considerations for **design** by helping sketch out the social setting of the system being developed. There is also **moderate-high** support for

	★★★	**implementation** because access control analysis can be used to bridge the gap between the security requirements and their implementation
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★	There is **low** support for testing, as it helps anticipate certain attackers which could be translated into possible test cases
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★	**Low**.
LSS5. Level to which the security requirements help reduce the overall development effort	★★★	Provides a **moderate-high** level of support for reducing the effort in developing the system, because aside from identifying possible attacks for the system, it also considers aspects like "why" they are relevant as well as countermeasures for them
LSS6. Support for other types of non-functional requirements (besides security) in later stages	★	There is **low** support for other types of non-functional requirements as only privacy is mentioned as a possible candidate

Table 6. i* Agent-based Requirements for Security Framework Application

6.7 Common Criteria

The Common Criteria (CC) is a formal method of requirement specifications for security standards [Sto01]. It was adopted in 1999 as an international standard for security product evaluation for establishing confidence in security. CC is a repeatable methodology for documenting Information Technology (IT) security requirements, documenting and validating product security capabilities, and promoting international cooperation in the area of IT security [Vet02]. CC can be used not only by software developers, but evaluators and consumers as well in their respective tasks.

The CC is mainly used to create two kinds of documents, a "protection profile" (PP) and a "security target" (ST) [CCIB99]. A PP is a document used to identify the desired security properties of a product created by a group of users. It is a list of user security requirements, described in a very specific way defined by the CC. A ST is a document that identifies what a product actually does, or a subset of it, that is security-relevant. An ST doesn't need to meet the requirements of any particular PP, but an ST could meet the requirements of one or more PPs. The PP and the ST are established through the development of three aspects: security environment, security objectives and security requirements [War06, Abr98, Kam05].

6.7.1 Framework Application

Table 7 shows the results of applying our framework to Common Criteria.

Common Criteria

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★	**Low.** While Common Criteria has been mainly designed to evaluate already elicited security requirements and literature suggests that there is no support for elicitation, we believe that certain aspects of elicitation could be accomplished with the use of Common Criteria
RE2. Type of elicitation technique used/recommended by the approach	⃠	**None.** While there is no explicit elicitation technique suggested, we believe that structured interviews could be used
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★	There is a **moderate** degree of identification of stakeholders. Common Criteria can help identify customers, developers, and evaluators needed for the process to be successful
RE4. Level of involvement of	★	There is **low** involvement from the

the customer (How involved in the elicitation process should the customer be)		customer, as most of the work is carried out by the developers and security experts
RE5. Elicitation of other types of requirements besides security	**NF** ★	**Non-functional requirements**. There is minimal (**low**) support for possibly eliciting privacy requirements
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	**S**	Security requirements would be elicited **sequentially**
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	⊘	**Not Supported**

REQUIREMENTS ANALYSIS

RA1. Type of overall analysis	**I** ★★★★ **+** **E** ★★★★	Common Criteria provides **both**, **internal and external** analysis of the security requirements elicited with a **high** degree of support for both
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★★★★	**High.** Common Criteria analyzes security requirements structurally and systematically to resolve any possible problems related to ambiguity
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★★★	**Moderate-high**. The formality of Common Criteria helps in ensuring that the requirements that have been developed so far are as complete as possible
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★	**Moderate.** While the security requirements are very specific, clarity can often be obscured by the Common Criteria's complexity and formality
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★★	**Moderate**

RA6. Analysis helps prevent security requirements conflict	★★★	Common Criteria can **definitely** aid in resolving conflicts between requirements; this mainly through their systematic approach and available tool support
REQUIREMENTS SPECIFICATION		
RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	**Va** ★★ + **Ve** ★★	Both **validation** and **verification** of the system could be done **moderately** with the security specifications produced by the Common Criteria
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	**C** ★	While no explicit mention to either time nor cost estimation is made in the Common Criteria literature, we believe that its formality and variety of artifacts produced could potentially aid in estimating the **cost** of the system; this is a **low** support though
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★★	The specification produced is **moderately** traceable; this because Common Criteria provides support for tracing which security requirements address which specific security objectives through a security requirements "rationale"
RS4. Consistency degree of the specification produced	★★★★	**High**. Besides the rigorous and systematic approach to specifying security requirements with the Common Criteria that ensures they are as consistent as possible, an APE criteria is also available to help evaluate their consistency
RS5. Support for specifying non-functional requirements other than security ones	⃠	**Not Supported.** While it might be possible, there is no explicit evidence that points to any support to specify non-functional requirements other than security
RS6. Overall clarity and understandability of the requirements specification	★★	While the level of formality of the Common Criteria makes its specifications **moderately** clear, the understandability suffers a little for those not trained in the Common Criteria process
RS7. Level of formality of the specification	**F**	The specifications produced are **formal**

RS8. Rigor of the specification process (how formal is the process itself)	F	The process is very **formal**
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★★★	**High.** The specifications produced using the Common Criteria can be extremely secure, as long as the approach is effectively applied (which can prove to be challenging)
REQUIREMENTS MANAGEMENT		
RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★	We believe that updating security requirements developed using the Common Criteria could prove a **difficult** task; this because the process is extremely formal and assessing the impact that updating a security requirement has on other ones can also prove complex
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	★★	Common Criteria provides **moderate** support for evolution; the approach is flexible enough to help foresee changes and accommodate for future evolution**.**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	★	**Low.** Many aspects of the process can potentially be automated
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★	It is very **difficult** to learn how to effectively apply the Common Criteria approach to a set of requirements. Besides its formality and complexity, developers must strengthen their backgrounds in a variety of aspects to effectively apply the approach
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★★★	The systematic nature of Common Criteria makes it **highly** scalable to much larger systems
RM6. Information availability regarding the approach (How popular is this approach?)	★★★	There is **abundant** information available regarding Common Criteria and its applications

LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	🛇	**None.** There is no explicit support/guidance for making the security requirements usable later on**.**
LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	**D** ★★ **+** **I** ★	Constraints for both **design** and **implementation** could be considered with the Common Criteria. There is **moderate** support for examining possible design representations of the TOE (Target of Evaluation). Additionally, the approach develops a security target (ST) which is an implementation-dependent statement of the security needs for a specific TOE (**low** support)
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★	**Low.** Some of the testing effort in the TOE could be used to help develop a testing strategy
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★★	The Common Criteria's main focus is evaluating security requirements to determine their level of security as well as to aid in improving it; certainly the artifacts developed with this focus in mind could be **moderately** helpful during testing
LSS5. Level to which the security requirements help reduce the overall development effort	★★★	Common Criteria can **moderately-highly** help reduce the development effort by ensuring that the security requirements have been developed as good as possible; this provides developers with a strong basis to continue the project from
LSS6. Support for other types of non-functional requirements (besides security) in later stages	🛇	**None**.

Table 7. Common Criteria Framework Application

6.8 SQUARE

The Security Quality Requirements Engineering (SQUARE) methodology [Mea05], created by the Software Engineering Institute's CERT Program, consists of nine steps that generate a final deliverable of categorized and prioritized security requirements. Although the SQUARE methodology could likely be generalized to any large-scale design project, it was designed for use with information technology systems.

The SQUARE process is most effective when conducted with a team of requirements engineers with security expertise and the stakeholders of the project [Gor05]. It begins with the requirements engineering team and project stakeholders agreeing on technical definitions that serve as a baseline for all future communication. Next, business and security goals are outlined. Then artifacts and documentation are created, which are necessary for a full understanding of the relevant system. Lastly, a structured risk assessment is performed to determine the likelihood and impact of possible threats to the system.

Following this work, the requirements engineering team determines the best method for eliciting initial security requirements from stakeholders. Once a method has been established, the participants rely on artifacts and risk assessment results to elicit an initial set of security requirements. Two subsequent stages are devoted to categorizing and prioritizing these requirements for management's use in making tradeoff decisions. Finally, an inspection stage is included to ensure the consistency and accuracy of the security requirements that have been generated [ChD04, Mea07].

6.8.1 Framework Application

Table 8 shows the results of applying our framework to SQUARE.

SQUARE

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★	SQUARE provides **low** support for eliciting security requirements.
RE2. Type of elicitation technique used/recommended by the approach	**Ot**	**Other.** This method helps you determine the best type of technique for the project at hand; these can range from interviews to surveys.
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★★	The approach helps identify a **moderate-high** degree of stakeholders; these can be customers, developers, or actors of the actual system.
RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	★★★★	The customers are **highly** involved in the elicitation process; they are entrusted with important tasks like artifact development for example

RE5. Elicitation of other types of requirements besides security	F ★ + NF ★	This approach is special because it not only looks at security as a **non-functional requirement** but as a **functional one** to better help adapt them; thus providing a **low** degree of support for considering other types of requirements
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	I	**Iterative**.
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★	**Low**.
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	I ★★★ + E ★★★	**Both external** and **internal** analysis of the security requirements elicited is provided with a **moderate-high** degree of support. The stakeholders and developers jointly help verify that the security requirements meet the security goals; this same approach is used to validating the security requirements
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★★★★	**High.** The approach urges for requirements to be as unambiguous as possible. It provides a dedicated step (requirements inspection) for resolving any ambiguity issues in them
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★★	**Moderate.**
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★★★	**High.** The high-customer involvement helps in clarifying the security requirements as much as possible. In addition, developers and customers agree on a common set of terminology and definitions to ensure that their communication is as clear as possible
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★	**Low.** The approach could potentially help consider alternative security requirements by the developers suggesting additional categories during the categorization steps

RA6. Analysis helps prevent security requirements conflict	★★★	**Definitely**. SQUARE during its security goals identification step helps align the stakeholders' views and interests; which in turn can help prevent conflicts from happening

REQUIREMENTS SPECIFICATION

RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	Ve ★★★	SQUARE strongly supports the idea of specifying only those security requirements that are "specific" enough that could be used to help **verify** the implemented system (**moderate-high** support)
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	C ★★★★	**Cost.** One of the stakeholder responsibilities is to ensure that the security requirements being specified are financially sound. This can help tremendously in estimating the costs of development; **high** support is thus provided
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★	Traceability of the specification is **low**
RS4. Consistency degree of the specification produced	★★★	A **moderate-high** level of consistency is provided. Security requirements inspections also set time aside to check for the consistency of the specifications
RS5. Support for specifying non-functional requirements other than security ones	★	**Low.** Safety requirements could potentially be specified with SQUARE; but no explicit process is described
RS6. Overall clarity and understandability of the requirements specification	★★	**Moderate**.
RS7. Level of formality of the specification	S	The specifications are **semi-formal**
RS8. Rigor of the specification process (how formal is the process itself)	S	The approach is **semi-formal**. While the activities are relatively informal, the process itself is well defined and formalized
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★★	**Moderate-high.** SQUARE helps produce highly secure specifications. This is mainly achieved through the level of involvement of the customer, and the quality of communication between them

		and the developers to ensure that issues that are identified as security problems are all addressed
REQUIREMENTS MANAGEMENT		
RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★★	**Moderate.**
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	★★★	**Moderate-high.** The approach can help in "steering" future improvements and modifications to the system. Security requirements can be developed with this in mind in order to accommodate evolution
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	★	**Low.** There is future plans for developing a tool that aids in the automation of the documentation
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★★	Learning is **moderately** difficult. While the approach itself is relatively simple, learning difficulties might be added depending on the techniques chosen along the process
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★★★	Scalability is **high**. The approach can likely be applied to projects of varying sizes
RM6. Information availability regarding the approach (How popular is this approach?)	★★★	**Abundant.**
LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	⃠	**None.** There is no explicit support described for making the security requirements developed usable at later stages of development

LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	A ★	Some **architectural** constraints can be identified with this approach (low support)
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★	**Low**. The prioritized categories could be used in order to help organize the test cases produced
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★	**Low.**
LSS5. Level to which the security requirements help reduce the overall development effort	★★	**Moderate.**
LSS6. Support for other types of non-functional requirements (besides security) in later stages	⃠	**None**. While safety requirements could be specified with this approach, there is no evidence that suggests any type of support past requirements engineering

Table 8. Common Criteria Framework Application

6.9 OCTAVE

The Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) is an information security risk evaluation approach that is comprehensive, systematic, and context driven [Alb05, Alb02]. Through the following of the OCTAVE Method, an organization can make information-protection decisions based on risks to the confidentiality, integrity, and availability of critical information technology (IT) assets. The operational units and the IT department of an organization work together to address the information security needs of the organization.

Using a three-phase approach, OCTAVE examines organizational and technology issues to assemble a comprehensive picture of the information security needs of an organization. The phases include,
- Phase 1: Build Asset-Based Threat
- Phase 2: Identify Infrastructure Vulnerabilities
- Phase 3: Develop Security Strategy and Plans

One of the advantages of the OCTAVE Method is that it is self-directed [Alb02]. A small team of the organization's personnel comprised of the operational units and the IT department of an organization becomes the analysis team; they manage the process and analyze all information [Alb99, Ric07].

6.9.1 Framework Application

Table 9 shows the results of applying our framework to OCTAVE.

Octave

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★★★★	Octave provides **high** support for security requirements elicitation
RE2. Type of elicitation technique used/recommended by the approach	**W**	**Workshops** are the primary technique used for elicitation**.** The organization's staff is brought to a series of knowledge-elicitation workshops
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★★★	**High**. Besides identifying relevant developers and customers, the approach helps identify levels of customers like senior management level, staff level, and operational areas level.
RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	★★★★	There is **high** involvement from the customer, because they are key in eliciting not only the security requirements, but also possible areas of concern, important assets, and current protection strategies; information that is

		essential for the effectiveness of Octave
RE5. Elicitation of other types of requirements besides security	**NF** ★★	**Moderate** support for other **non-functional requirements** like confidentiality and integrity requirements elicitation is provided
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	**I**	Security requirements are elicited **iteratively**, through a series of workshops that help refine what the customers want
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★★	**Moderate**.
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	**I** ★★★★	**High internal** analysis of the security requirements elicited is performed by the analysis team; this serves to verify them
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	⃠	**None.** There is no clear support provided for detecting and/or resolving unambiguity issues in the security requirements
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★★	**Moderate** support for completeness assurance can be obtained from the variety of stakeholders identified.
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★★	**Moderate-high.** Octave strives for clear security requirements; to aid in this the security requirements (threat profiles) can support visual representations to aid in ensuring they are as clear as possible
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★	**Low** support is explicit for alternative requirements consideration; although the different threat categories could be used to determine missing security requirements that are needed to secure them
RA6. Analysis helps prevent security requirements conflict	★	**Low**. While the stakeholders can help in dealing with requirements conflict, there is no mention of steps that developers can take in order to identify conflicts and resolve them

REQUIREMENTS SPECIFICATION

RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	**Va** ★★	The threat profiles can be used to **moderately validate** that the system has effectively adopted the necessary measures to mitigate the threats described
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	**C** ★★★★	A detailed **cost**-benefit analysis can be performed in order to determine the financial implications of certain specified security requirements (**high** support)
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★★	**Moderate**.
RS4. Consistency degree of the specification produced	★	**Low**. Could not find much evidence of any support for guaranteeing that the resulting specifications are consistent, although the process to specifying the security requirements suggests that the resulting specification would be consistent
RS5. Support for specifying non-functional requirements other than security ones	⃠	**Not Supported.**
RS6. Overall clarity and understandability of the requirements specification	★★★	The overall clarity and understandability of the security requirements specification is **moderate-high**
RS7. Level of formality of the specification	**S**	The specifications produced are **semi-formal**
RS8. Rigor of the specification process (how formal is the process itself)	**F**	The process of the approach is **formal**
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★	**Moderate.** The specification helps in determining what steps are needed in order to effectively secure the requirements

REQUIREMENTS MANAGEMENT

RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★	**Difficult**. Not only do you have to concentrate the specific requirement that needs to be updated, you must also consider the threat profiles and possible risks that are associated with them

RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	⃠	**Not Supported.**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	⃠	**None.**
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★	It is **difficult** to learn how to effectively apply Octave; this because background knowledge on a variety of aspects like risk assessment and information infrastructure are also necessary
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★★★	**High**. Octave can easily be adapted to cover small or large projects; specific methods like Octave and Octave-S are specifically designed for each one
RM6. Information availability regarding the approach (How popular is this approach?)	★★★	There is **abundant** information available
LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	★	**Low.** Could potentially help during implementation where the security requirements can be integrated with the security strategy developed
LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	⃠	**None.**
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★★	There are **moderate** benefits for testing. Among what is elicited from the customers are the areas of concern; these can be used as plausible scenarios of what could go wrong with the system, and therefore define test cases that embody those scenarios

LSS4. Degree of support of the overall focus of the approach when it comes to testing	★★	**Moderate.**
LSS5. Level to which the security requirements help reduce the overall development effort	★★	**Moderate**. While a variety of the artifacts produced could be used later on; there are not a whole lot of extra things obtained from the requirements engineering process that could aid later on
LSS6. Support for other types of non-functional requirements (besides security) in later stages	⃠	**None**.

Table 9. OCTAVE Framework Application

6.10 Attack Trees

According to Brice Schneier [Sch00], “attack trees provide a formal, methodical way of describing the security of systems, based on varying attacks.” The notion of attack trees [Wel03, Whi01] has been considered as a method for modeling attacks. Attack trees have been described recently as a systematic method to characterize system security based on varying attacks [Vie01]. Several approaches for security requirements are based on trees. Attack Trees are used in development of intrusion scenarios, which can then be used to identify requirements.

Attacks against a system are represented using a tree structure, with the goal of attacking the system as the root of the tree and different ways of achieving that goal as leafs of the tree. Attack trees refine information about attacks by identifying the compromise of enterprise security or survivability as the root of the tree. Each path through an attack tree represents a unique attack on the enterprise [Moo1]. Each attack tree enumerates and elaborates the ways that an attacker could cause the event to occur. Each path through an attack tree represents a unique attack on the enterprise [12].

6.10.1 Framework Application

Table 10 shows the results of applying our framework to the attack trees approach.

Attack Trees

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★★★	Attack Trees provide a **moderate-high** degree of support for eliciting security requirements. This is accomplished by modeling the steps/actions that it would take for a certain attack to be accomplished
RE2. Type of elicitation technique used/recommended by the approach	**I + Ot**	**Interviews and Other**. The security requirements are elicited by obtaining the possible attacks on the system to be developed from the customer, and negating the leaf nodes that are produced in the attack tree; these become the security requirements
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★	There is **low** identification of stakeholders; although attack trees can also help identify assets and attackers
RE4. Level of involvement of	★	**Low**. The customer is only really needed

the customer (How involved in the elicitation process should the customer be)		to help provide some of the attacks that the system under development could be susceptible to
RE5. Elicitation of other types of requirements besides security	**NF** ★★★	Other **non-functional requirements** like **privacy and safety** could also be elicited with attack trees; there is **moderate-high** support for eliciting both with attack trees
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	**S**	Security requirements are elicited **sequentially**; meaning that developers create the attack tree from a top down approach, and there is not much of an iteration to adjust them if needed
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★	Attack trees are so focused on eliciting security requirements based on specific attacks, that they provide **low** support for defining the scope of the system
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	**I** ★★	The overall analysis is mainly **internal**; this analysis is **moderate**
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★★	**Moderate**. The unambiguity resolution is mostly dependant on the depth of the attack tree
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	⃠	**None**. There is no explicit information of any attempt to ensure that the security requirements are complete
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★	**Moderate**. Clarity also depends on how deep the attack tree is (i.e. the deeper the tree, the clearer the security requirements because the statements become simpler)
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★★★★	**High.** An attack tree can theoretically help consider many, if not all, the possible ways of achieving a certain attack; this helps in considering alternative and additional security requirements for mitigating all the attack possibilities

RA6. Analysis helps prevent security requirements conflict	★★	**Marginally**. Though the attack tree structure can help you spot conflict problems, no real resolution technique is provided
REQUIREMENTS SPECIFICATION		
RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	Va ★★ + Ve ★★	The specification produced could support **both** the **validation and verification** of the resulting system. Techniques like risk analysis, reliability analysis, and shortest path analysis can very well be applied to an attack tree in order to help verify (**moderate**) and validate (**moderate**) a system
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	C ★★ + T ★★	Specifications produced with attack trees can help with both **cost and time** estimations if meaningful values are assigned to the nodes of the attack tree. **Moderate** support for both of these as attack trees do not support these features naturally, but they can be adapted to do so
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★★★	**Moderate-high**. The graphical nature of attack trees allows for easy tracing of not only the security requirements, but also to other aspects like goals and attacks
RS4. Consistency degree of the specification produced	★	**Low**.
RS5. Support for specifying non-functional requirements other than security ones	★★	**Moderate.** Attack trees could be adapted to also help specify safety and privacy requirements.
RS6. Overall clarity and understandability of the requirements specification	★★	**Moderate**. The overall clarity and understandability of attack trees could potentially be higher if the trees are expanded deep enough
RS7. Level of formality of the specification	I	The specification produced is **informal**, because, aside from the tree structure, the trees are usually filled using natural language
RS8. Rigor of the specification process (how formal is the process itself)	S	The approach is **semi-formal**; they provide their own structure and syntax but still rely on natural language
RS9. Overall security level of the resulting system (How	★	**Low.** The problem with attack trees is that they depend too much on the

secure can the resulting system be based on the specifications?)		expertise of the security analyst and developers producing them. Additionally, there is no standard way of either constructing them or analyzing them.

REQUIREMENTS MANAGEMENT

RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★★★	**Moderate-easy**. Given the simplicity of attack trees updating them should not be that difficult; also, their graphical nature helps identify other aspects that could be affected by updates
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	⃠	**Not Supported.**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	★	**Low.** While there is no mention about automation of attack trees, we believe that a variety of steps in their process could be easily automated
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★★★★	While to make them as effective as possible a lot depends on your expertise on security attacks, attack trees themselves are **easy** to learn and relate to
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★	**Low**. Attack trees could become extremely complex and ultimately useless if applied to large systems
RM6. Information availability regarding the approach (How popular is this approach?)	★★	There is **moderate** level information available.

LATER STAGES SUPPORT

LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	⃠	**None.** While attack trees could be used in architecture risk analysis, no real integration support is provided

LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	D ★★	Attack trees can support **design** decisions in order to mitigate attacks **moderately.**
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★★	**Moderate.** Since attack trees help focus on measurable goals, this information can be used to create specific test cases that test for a certain attack to happen
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★★★	**Moderate-high**. The focus of attack trees is decomposing an attack to the point that independent events are considered that constitute the whole attack; this focus can be directly translated to the testing effort to test for situations described in the "leaf" nodes of the tree
LSS5. Level to which the security requirements help reduce the overall development effort	★	**Low**. Countermeasures can be easily derived from attack trees; this information can help reduce the amount of implementation options for the security measures
LSS6. Support for other types of non-functional requirements (besides security) in later stages	⦸	**None**. There is no explicit information available regarding support for other types of non-functional requirements past requirements engineering stages

Table 10. Attack Trees Framework Application

6.11 USeR Method

The USeR method [HaH06] is directed towards extracting security issues based on statements made by the customers and potential users about the system to be developed. Once these security issues have been extrtacted, USeR assists in determining the specific security needs (requirements) for the system and their relationships to potential technical security solutions. The USeR method prescribes that the process of determining these security requirements should happen in parallel with determining other requirements of the system (functional and/or non-functional).

The USeR method is aimed at increasing the involvement of security experts during the requirements stage of development; it accomplishes this by basing itself on the tools and principles of Quality Function Development (QFD). [Aka97, Cha02, Coh95]. The USeR method is based on 5 main steps,

1- Identify security-related statements

2- Determine security needs

3- Determine security requirements

4- Determine security techniques

5- Explore design implications

6.11.1 Framework Application

Table 11 shows the results of applying our framework to the USeR method.

USeR

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★★★★	The USeR method provides a **high** degree of elicitation of security requirements; this is mostly accomplished by decomposing design documents and extracting/eliciting possible security requirements from them
RE2. Type of elicitation technique used/recommended by the approach	**W + Ot**	**Workshops** are the primary technique used for elicitation along with structured meetings**.**
RE3. Degree of stakeholder identification provided	★★★	**Moderate-high**. The approach helps identify user representatives, developers,

(Including customer, developers, end-users)		customers, as well as security experts
RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	★★★★	There is **high** involvement from the customer in the elicitation process
RE5. Elicitation of other types of requirements besides security	**NF** ★★★	Other **non-functional requirements** by identifying other possible categories of need like privacy and security (**moderate-high** support)
RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	**S**	**Sequentially**. Security requirements are elicited through following a series of 5 steps
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★	**Low**. There is some support that could help identify possible limitations of the system; information that could be used to determine what is outside of the system's scope
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	**E** ★★	A **moderate external** analysis of the security requirements elicited is performed
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★★★	**Moderate-high.** The elicited security needs and security requirements resulting from them are constantly checked by the customers and developers in order to ensure unambiguity
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★★★★	**High.** Hierarchical diagrams are used to help detect missing/incomplete security requirements.
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★	**Moderate.** The design team (customers and developers) get together to ensure the requirements are clear to both parties
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★★★	**Moderate-high.** The approach supports the consideration of additional as well as missing security requirements. Support is explicit for alternative requirements

		consideration; although the different threat categories could be used to determine missing security requirements that are needed to secure them
RA6. Analysis helps prevent security requirements conflict	★★★	**Definitely**. Time is dedicated to checking for possible conflicts and bringing the involved parties together to help resolve them
	REQUIREMENTS SPECIFICATION	
RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	**Va** ★	The specification produced could help **validate** the resulting system with **low** support
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	⃠	**None.**
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★★★★	**High**. A matrix system is used to allow for traceability of the security requirements to security needs as well as security requirements to security techniques
RS4. Consistency degree of the specification produced	★★★	**Moderate-high**. The specifications produced are consistent with the added support of pair-wise comparison by security experts
RS5. Support for specifying non-functional requirements other than security ones	★★★	**Moderate-high.** Provides support for identifying other possible categories like privacy and trust; these could also be used to help specify requirements for them
RS6. Overall clarity and understandability of the requirements specification	★★	**Moderate.**
RS7. Level of formality of the specification	**S**	**Semi-formal**
RS8. Rigor of the specification process (how formal is the process itself)	**F**	The process for specifying the security requirements is **formal** as it is very structured and systematic
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★	**Moderate.**

REQUIREMENTS MANAGEMENT		
RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★★★	Security requirements can be **easily** updated due to the high degree of traceability provided in the specification.
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	★	**Low.**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	⊘	**None.**
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★★★★	**Easy.**
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★	Scalability is **moderate**. Applying this approach to large systems can prove challenging, because in large projects it is much more difficult to bring so many stakeholders together and this approach depends a lot on that aspect
RM6. Information availability regarding the approach (How popular is this approach?)	★	**Low.** There is not a lot of information regarding this approach.

LATER STAGES SUPPORT		
LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	★	**Low.** There is a possibility of integration with design and possibly implementation
LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	**D** ★★ + **I**	Some **design** considerations could be developed from the security requirements as possible design implications are explored as a major step of the method. Also, **implementation**

	★	aspects are also considered during the step where possible technologies are explored
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	🛇	**None**.
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★	**Low.**
LSS5. Level to which the security requirements help reduce the overall development effort	★★★	**Moderate-high**. Security techniques are developed in order to operationalize the security requirements. In addition, the method also explores current security technology that could be used to implement the security requirements
LSS6. Support for other types of non-functional requirements (besides security) in later stages	★	**Low**. Requirements related to privacy and trust could potentially be considered during design

Table 11. USeR Method Framework Application

6.12 CLASP

CLASP (Comprehensive, Lightweight Application Security Process) [Vie05] provides a well-organized and structured approach to moving security concerns into the early stages of the software development lifecycle. CLASP is a set of process pieces that can be integrated into any software development process. It is designed to be both effective and easy to adopt. It takes a prescriptive approach, and documents activities that organizations should be doing. In turn, it provides a variety of security resources that make implementing those activities reasonable.

The core of CLASP is made of thirty new activities that can be integrated into a software development process. The initial section of the activities belongs to the project manager. While those duties do not constitute a significant time commitment, they do reflect the CLASP philosophy that effective security practices require organizational "buy-in." [SSI05, Tar95]

6.12.1 Framework Application

Table 11 shows the results of applying our framework to CLASP.

CLASP

REQUIREMENTS ELICITATION		
RE1. Degree of support for requirements elicitation	★★	CLASP provides **moderate** support for eliciting security requirements
RE2. Type of elicitation technique used/recommended by the approach	I + Ot	**Interviews and Other.** This method mainly uses existing document reviews to elicit security requirements
RE3. Degree of stakeholder identification provided (Including customer, developers, end-users)	★★★★	**High** degree of stakeholders identification provided by CLASP. The approach helps identify important people needed for each stage of the process as well as it assigns specific roles to them, like auditors, specifiers, developers, architects, etc.
RE4. Level of involvement of the customer (How involved in the elicitation process should the customer be)	★	There is a **low** degree of involvement of the customer in the elicitation process; they mainly help in eliciting misuse cases for certain aspects of the process
RE5. Elicitation of other types of requirements besides security	F ★★	This approach helps identify **functional requirement** as well as **non-functional** aspects of the system as they relate to the
RE5. Elicitation of other types of requirements besides security	F	This approach helps identify **functional requirement** as well as **non-functional**

RE6. Dynamics of the elicitation process (i.e. Iteration of requirements elicitation or not)	S	**Sequential**.
RE7. Support for establishing system boundaries (What is inside/outside the scope of the system being developed)	★★	**Moderate** support for determining the scope of the system; it also helps identify “trust” boundaries of the system as well as the resources that would be needed and those that would be outside of the scope of the system
REQUIREMENTS ANALYSIS		
RA1. Type of overall analysis	I ★★★	**High internal** analysis of the security requirements elicited is provided. The approach identifies a “requirements tester” that helps in verifying the elicited security requirements
RA2. Unambiguity resolution level of the analysis (Unambiguity issues can be detected and resolved through the analysis)	★★	**Moderate.**
RA3. Completeness resolution level of the analysis (Analysis can help determine if the security requirements are complete)	★★	**Moderate.**
RA4. Clarity resolution level of the analysis (Analysis helps clarify the security requirements as much as possible)	★★	**Moderate.**
RA5. Support level of analysis to consider alternative/additional security requirements missed during elicitation	★★★★	**High.** The variety of problem types that have been identified (104 types of different vulnerabilities) can be used in order to consider alternative and additional security requirements that have been missed during elicitation but are needed to address the vulnerabilities
RA6. Analysis helps prevent security requirements conflict	★★★	**Definitely**. SQUARE during its security goals identification step helps align the stakeholders’ views and interests; which in turn can help prevent conflicts from happening

RA6. Analysis helps prevent security requirements conflict ★★★ **Definitely**. SQUARE during its security goals identification step helps align the

REQUIREMENTS SPECIFICATION

RS1. The specification produced can be used as a baseline for system validation and/or verification once implemented	**Va** ★★ **+** **Ve** ★	CLASP specifications could be used for both **validation** (**moderate**) and **verification** (**low**) of the system
RS2. The specification provides a basis for cost and/or time estimation of the overall development project	**C** ★	The **cost** of implementing certain activities can be explored in the specifications (**low** support)
RS3. Traceability level of the specification produced (How traceable is the security requirements specification)	★	Traceability of the specification is **low**
RS4. Consistency degree of the specification produced	★★	A **moderate** level of consistency is provided. Metrics are defined by the approach to help “standardize” the communication between the stakeholders, this in turn reflects in the consistency of the specifications
RS5. Support for specifying non-functional requirements other than security ones	★★★	**High.** There is support for specifying other functional and non-functional requirements as well as relating them to the success of the specified security requirements (helps you specify the functional and non-functional mechanisms needed to implement the security requirements)
RS6. Overall clarity and understandability of the requirements specification	★★	**Moderate.**
RS7. Level of formality of the specification	**S**	**Semi-formal**.
RS8. Rigor of the specification process (how formal is the process itself)	**F**	**Formal**.
RS9. Overall security level of the resulting system (How secure can the resulting system be based on the specifications?)	★★★★	CLASP helps develop **highly** secure specifications as well as extensive support for assessing the level of security
RS9. Overall security level of the resulting system (How	★★★★	CLASP helps develop **highly** secure specifications as well as extensive

REQUIREMENTS MANAGEMENT

RM1. Level of difficulty for updating security requirements (i.e. making additions, deletions, and/or modifications)	★	Updating the security requirements can be a **difficult** task**.** The security requirements can be very intricate depending on the specific system that they are for; additionally since there is a lot of other aspects that the requirements are associated with, it would also be difficult to update these associations
RM2. Level of security requirements evolution supported (As the system evolves, does the approach support for the requirements to evolve as well?)	★	**Low.**
RM3. Level of automation provided (Is there any support for automating any step and/or process in the approach)	★★★	**Moderate-high.** This approach consciously makes itself flexible enough to allow tools (internal and external) to help automate a variety of the steps of the process
RM4. Degree of learning difficulty of the approach (How difficult is it for a novice user to learn this approach?)	★	Learning can be **difficult**. There are a lot of activities involved in the approach, as well as many conventions to learn
RM5. Scalability of the approach (Does the approach support relatively easy application to systems of various sizes?)	★★★★	The scalability degree of the approach is **high**. The recommendations and artifacts produced by the approach can be applied to a variety of systems; furthermore, the approach can be applied to either new systems or existing ones looking to "retrofit" security into the requirements
RM6. Information availability regarding the approach (How popular is this approach?)	★★★★	**High.** Beyond information about the approach from other sources, the approach itself has extensive documentation

LATER STAGES SUPPORT

LSS1. Support for integrating security requirements with later stages of development (How usable are the security requirements past their inception?)	★★	Provides **moderate** support for integrating security requirements because they can help guide the development team on how to apply the security requirements to the possible design. In addition, the approach covers different aspects for securing the system at later stages of the development cycle,
LSS1. Support for integrating	★★	Provides **moderate** support for

LSS2. Constraint consideration for later stages (Does the approach allow for planning other aspects of the system past the requirements?)	A★ + D★★ + I★ + M★	Helps consider early thoughts about **architecture** (**low**) with attack vectors support as well as support from architects when specifying the security requirements in the first place. For **design** (**moderate**), it provides consideration for the application of the security requirements and principles. It also helps consider different technologies available for the **implementation** (**low**) and future aspects of the **maintenance** (**low**) effort
LSS3. Security requirements provide testing benefits/support for later stages (Can the security requirements produced be used as a basis for testing the system?)	★★★★	**High.** The approach has a strong call for testability of the system based on the security requirements using "automated tests." The approach also provides a specific activity dedicated to identifying, implementing, and performing security tests of the system.
LSS4. Degree of support of the overall focus of the approach when it comes to testing	★★	The focus of the approach can be **moderately** applied to security
LSS5. Level to which the security requirements help reduce the overall development effort	★★★★	There is **high** support for reducing the development effort with the security requirements. On top of the support for the other stages of development (including testing) there is also support for identifying the most common vulnerabilities of the system; this information can be used to steer the development of the system in order to avoid them. There is also "vulnerability remediation procedures" that are developed in order to establish measures for mitigating the vulnerabilities.
LSS6. Support for other types of non-functional requirements (besides security) in later stages	★	**Low.**

Table 12. CLASP Method Framework Application

7 Surveyed Approaches Results Comparison

Once the framework was applied to each one of the 12 methods, the results of each question were organized into tables that allow for easy comparison of the results of all the approaches surveyed. Each comparison table includes a variety of information; it summarizes each question as columns and each approach surveyed as rows. The answers to each question are then summarized by each approach and a count of stars is kept. This star count is done at two level, approaches and questions. The total star count for each approach is presented as an added column; the star count for each question in the phase is presented as an added row. Keeping the star count for both approaches and questions allows us to observe trends not only in which approaches did better and worst, but also which questions have more support than others.

As described in the introduction, this survey contains 34 different questions. 30 of these questions are subjective in the sense that their response is rated (star rating), while 4 of them are just used for data collection. From this point on, we will focus mainly on those 30 questions that have a star rating, as most of the comparison and analysis of the results are based on star counts.

Tables 13 through 17 provide side-by-side comparison of the answers to each question categorized by each one of the 5 phases of security requirements engineering surveyed.

Requirements Elicitation								
	RE1 Elicit Support	**RE2** Elicit Tech	**RE3** Stakehldr ID	**RE4** Customr Involve	**RE5** Elicit Other Reqs.	**RE6** Elicit Dynam	**RE7** System Boundar ies	**Total Star Count (per approach)**
Misuse Cases	★★★★	**B**	★★	★★★	**NF**★★★	**I**	★★	14 Stars
Abuser Stories	★★★	**I**	★★★★	★★★	⃠	**I**	★	11 Stars
Secure TROPOS	★	**Ot**	★★	★	**NF** ★	**I**	★	6 Stars
Sec. Prob. Frames	⃠	⃠	⃠	⃠	⃠	⃠	⃠	0 Stars
Anti-Models	★★	**Ot**	★★★	★★	**NF** ★★★	**I**	★★★	13 Stars
i*	★★	**I** **Ot**	★★★★	★★★	**NF** ★	**I**	★	11 Stars
Common Criteria	★	⃠	★★	★	**NF** ★	**S**	⃠	5 Stars
SQUARE	★	**Ot**	★★★	★★★★	**F**★ **NF**★	**I**	★	11 Stars
Octave	★★★★	**W**	★★★★	★★★★	**NF**★★	**I**	★★	16 Stars
Attack Trees	★★★	**I** **Ot**	★	★	**NF**★★★	**S**	★	9 Stars
USeR	★★★★	**W** **Ot**	★★★	★★★★	**NF**★★★	**S**	★	15 Stars
CLASP	★★	**I** **Ot**	★★★★	★	**F**★★ **NF**★★	**S**	★★	13 Stars
Total Star Count (per question)	27 Stars	NO Stars	32 Stars	27 Stars	23 Stars	NO Stars	15 Stars	**124 Total Star Count**

Table 13. Security Requirements Elicitation Results Comparison

Requirements Analysis							
	RA1 Analysis Type	**RA2** Unambig. Resoltn	**RA3** Complete Resoltn	**RA4** Clarity Resoltn	**RA5** Alternative Requs	**RA6** Conflict Resoltn	**Total Star Count (per approach)**
Misuse Cases	I★★★ E★	★	★★	★	★★	★★★	13 Stars
Abuser Stories	I★	★	★	★★	⊘	⊘	5 Stars
Secure TROPOS	E★★	★	★★	★★	⊘	★★	9 Stars
Sec. Prob. Frames	E★★	★★★	★★★★	★★	★★★	★★	16 Stars
Anti-Models	I★	★★★	★	★★★	★★★	★★	13 Stars
i*	I★★	★	★★	★★★	★★	★★★	13 Stars
Common Criteria	I ★★★★ E★★★★	★★★★	★★★	★★	★★	★★★	22 Stars
SQUARE	I★★★ E★★★	★★★	★★	★★★★	★	★★★	20 Stars
Octave	I★★★★	⊘	★★	★★★	★	★	11 Stars
Attack Trees	I★★	★★	⊘	★★	★★★★	★★	12 Stars
USeR	E★★	★★★	★★★★	★★	★★★	★★★	17 Stars
CLASP	I★★★	★	★★	★★	★★★★	★★	16 Stars
Total Star Count (per question)	37 Stars	25 Stars	25 Stars	28 Stars	25 Stars	27 Stars	**167 Total Star Count for Analysis**

Table 14. Security Requirements Analysis Results Comparison

Requirements Specifications

	RS1 System Valid/ Verif	**RS2** Cost/ Time Estimate	**RS3** Trace Level	**RS4** Specs Consistent	**RS5** NonFunc Specs	**RS6** Clarity/ Underst	**RS7** Specs Formal	**RS8** Process Formal	**RS9** Resulting System's Security	**Total Star Count (pe approach)**
isuse ases	**Va** ★★ **Ve** ★★	**C**★★	★	★★	★★★	★★	**I**	**I**	★★	16 Stars
buser tories	**Va** ★	**C**★★ **T**★★★	★★★★	★	⦸	★★	**I**	**I**	★	14 Stars
ecure ROPOS	**Va**★★★	**C**★ **T**★	★	★★★	★★	★★	**S**	**F**	★★	16 Stars
ec. Prob. rames	⦸	**T**★★★	★	★★	★	★	**F**	**S**	★★	10 Stars
nti- odels	**Ve**★★	⦸	★★	★★	★★★★	★	**S**	**F**	★★	13 Stars
	Va★	**C**★★	★★	★★★★	★★★★	★★	**S**	**F**	★★★★	19 Stars
ommon riteria	**Va**★★ **Ve**★★	**C**★	★★	★★★★	⦸	★★	**F**	**F**	★★★★	17 Stars
QUARE	**Ve** ★★	**C**★★★ ★	★	★★★	★	★★	**S**	**S**	★★★	16 Stars
ctave	**Va**★★	**C**★★★ ★	★★	★	⦸	★★★	**S**	**F**	★★	14 Stars
ttack rees	**Va**★★★ **Ve**★★	**C**★★ **T**★★	★★★	★	★★★	★★	**I**	**S**	★	17 Stars
SeR	**Va**★	⦸	★★★★	★★★	★★★	★★	**S**	**F**	★★	15 Stars
LASP	**Va**★★ **Ve**★	**C**★	★	★★	★★★	★★	**S**	**F**	★★★★	16 Stars
otal Star ount er uestion)	28 Stars	28 Stars	24 Stars	28 Stars	23 Stars	23 Stars	NO Stars	NO Stars	29 Stars	**183 Total Star Count fo Specs**

Table 15. Security Requirements Specification Results Comparison

Requirements Management							
	RM1 Update Difficulty	**RM2** Requs. Evo	**RM3** Level of Auto.	**RM4** Learning Difficulty	**RM5** Scalability	**RM6** Info Availability	**Total Star Count (per approach)**
Misuse Cases	★★★★	⃠	★	★★★★	★	★★★	13 Stars
Abuser Stories	★★★★	⃠	⃠	★★★	★	★	9 Stars
Secure TROPOS	★★	⃠	★★★	★★	★★★★	★★	13 Stars
Sec. Prob. Frames	★★	★	⃠	★★	★★★	★★	10 Stars
Anti-Models	★★	⃠	★★	★	★★★★	★	9 Stars
i*	★	⃠	⃠	★★	★★	★	6 Stars
Common Criteria	★	★★	★	★	★★★★	★★★	12 Stars
SQUARE	★★	★★★	★	★★	★★★★	★★★	15 Stars
Octave	★	⃠	⃠	★	★★★★	★★★	9 Stars
Attack Trees	★★★	⃠	★	★★★★	★	★★	11 Stars
USeR	★★★	★	⃠	★★★★	★★	★	11 Stars
CLASP	★	★	★★★	★	★★★★	★★★★	14 Stars
Total Star Count (per question)	25 Stars	8 Stars	12 Stars	27 Stars	34 Stars	26 Stars	**132 Total Star Count for Management**

Table 16. Security Requirements Management Results Comparison

Later Stages Support

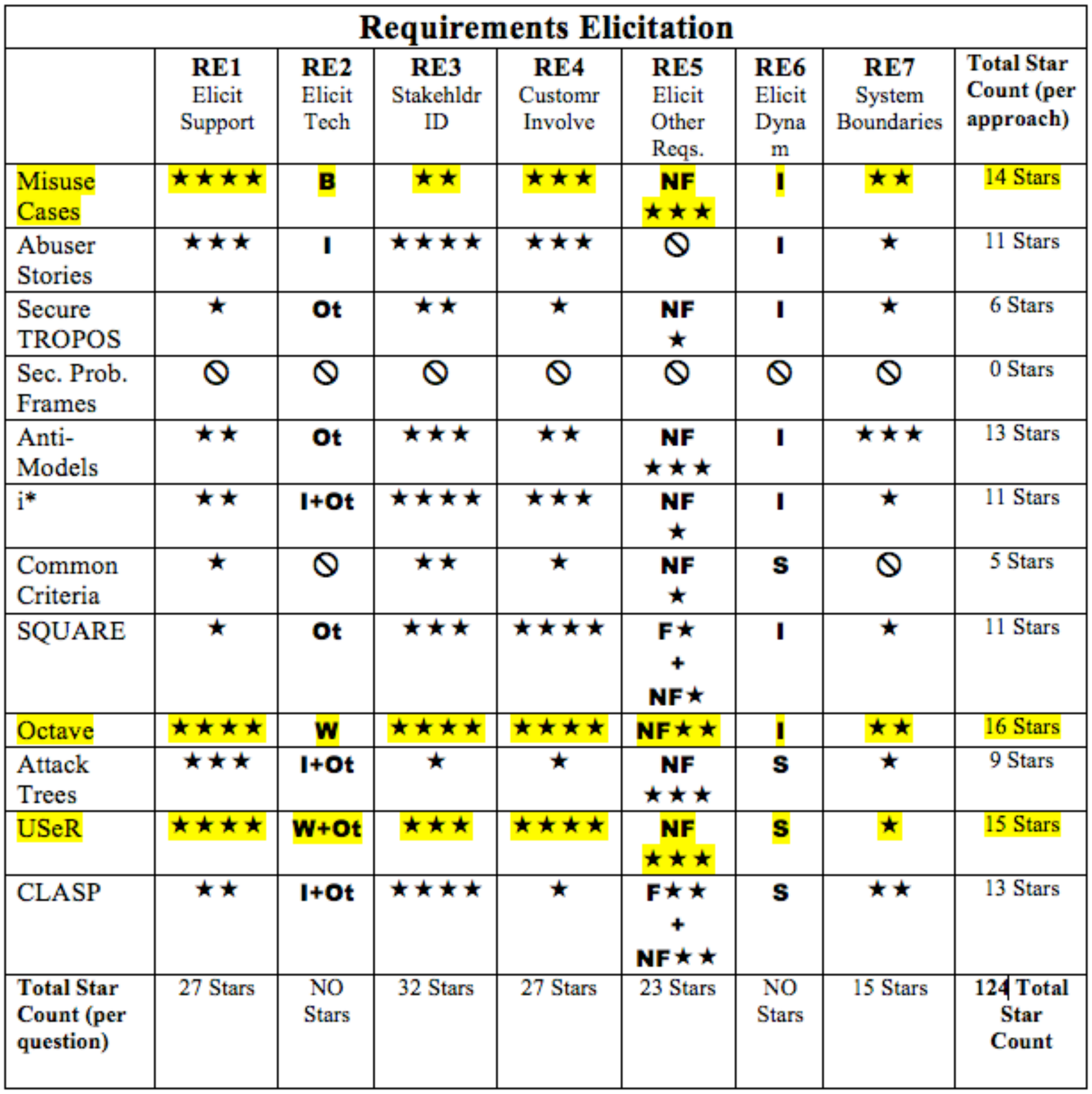
	LSS1 Sec. Requs. Integration	**LSS 2** Constraint Consider	**LSS3** Testing Benefits	**LSS4** Focus for Testing	**LSS5** Reduce Development Effort	**LSS6** Other Requs. Later	**Total Star Count (per approach)**
Misuse Cases	🛇	🛇	★★★	★★★	★★	★	9 Stars
Abuser Stories	🛇	I★★	★	★★	★★	🛇	7 Stars
Secure TROPOS	★★	A★★ D★★	★★	★	★	★★ ★	13 Stars
Sec. Prob. Frames	🛇	A★★★★	🛇	★	★★★	🛇	8 Stars
Anti-Models	🛇	🛇	★	★	★★	🛇	4 Stars
i*	★	D★★ I★★★	★	★	★★★	★	12 Stars
Common Criteria	🛇	D★★ I★	★	★★	★★★	🛇	9 Stars
SQUARE	🛇	A★	★	★	★★	🛇	5 Stars
Octave	★	🛇	★★	★★	★★	🛇	7 Stars
Attack Trees	🛇	D★★	★★★	★★★★	★	🛇	10 Stars
USeR	★	D★★ I★	🛇	★	★★★	★	9 Stars
CLASP	★★	A★ D★★ I★ M★	★★★★	★★	★★★★	★	18 Stars
Total Star Count (per question)	7 Stars	29 Stars	19 Stars	21 Stars	28 Stars	7 Stars	**111 Total Star Count for LSS**

Table 17. Later Stages Support for Security Requirements Results Comparison

7.1 Best Approaches per Phase

With the side-by-side comparisons of how well each one of the approaches performed at each phase and each question, the first aspect we were able to determine is the approaches that performed the best at each phase. The determining factor for the approaches that performed best was based on the total star count per phase; meaning that those approaches that had the higher star count provide the best support for that phase.

Figure 9 shows the best approaches found for the Elicitation phase; in this case we had a total of 4 approaches (misuse cases, Octave, USeR, and CLASP) that came out as best.

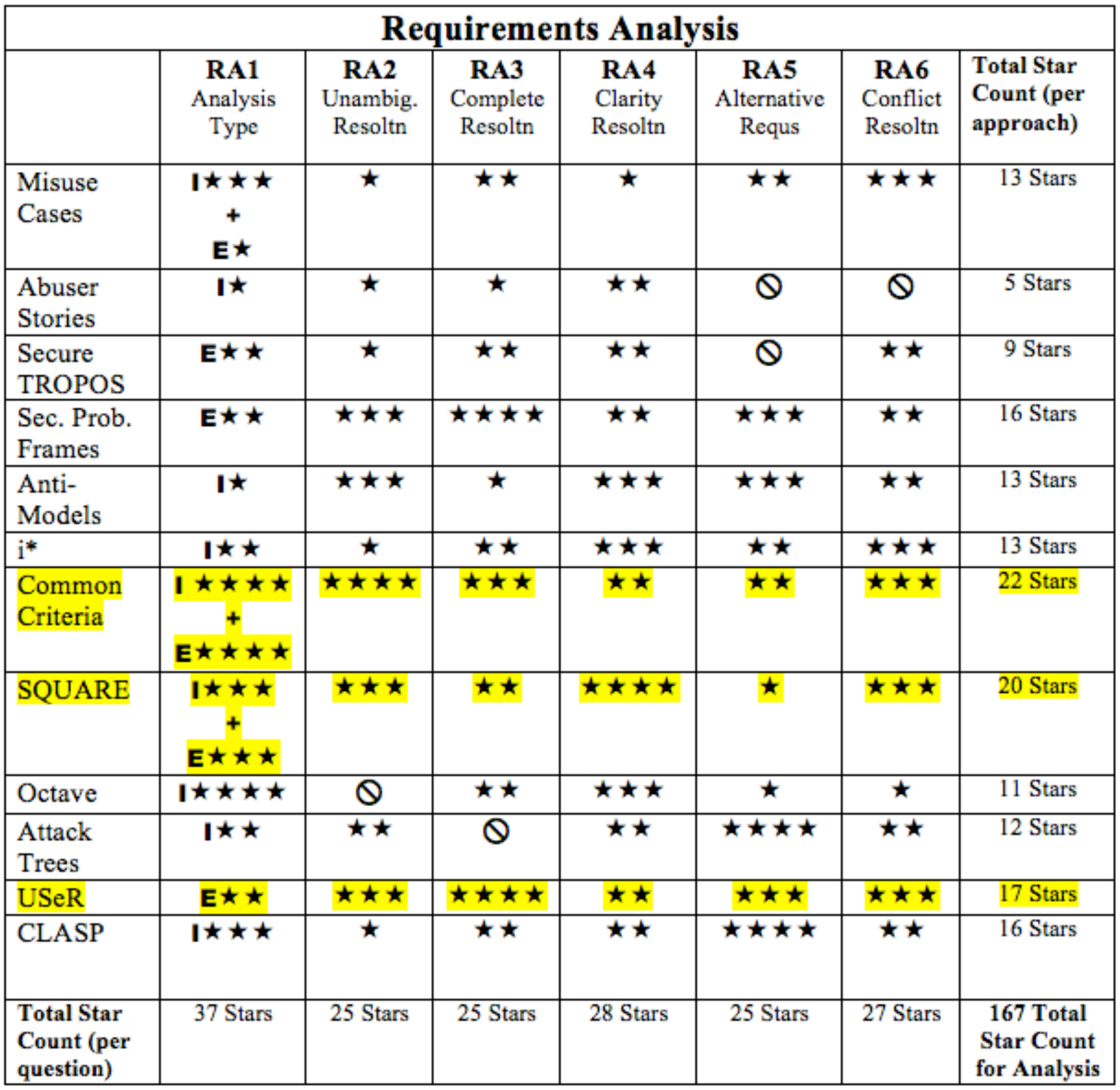


Requirements Elicitation

	RE1 Elicit Support	**RE2** Elicit Tech	**RE3** Stakehldr ID	**RE4** Customr Involve	**RE5** Elicit Other Reqs.	**RE6** Elicit Dynam	**RE7** System Boundaries	**Total Star Count (per approach)**
Misuse Cases	★★★★	**B**	★★	★★★	**NF** ★★★	**I**	★★	14 Stars
Abuser Stories	★★★	**I**	★★★★	★★★	⊘	**I**	★	11 Stars
Secure TROPOS	★	**Ot**	★★	★	**NF** ★	**I**	★	6 Stars
Sec. Prob. Frames	⊘	⊘	⊘	⊘	⊘	⊘	⊘	0 Stars
Anti-Models	★★	**Ot**	★★★	★★	**NF** ★★★	**I**	★★★	13 Stars
i*	★★	**I+Ot**	★★★★	★★★	**NF** ★	**I**	★	11 Stars
Common Criteria	★	⊘	★★	★	**NF** ★	**S**	⊘	5 Stars
SQUARE	★	**Ot**	★★★	★★★★	**F**★ + **NF**★	**I**	★	11 Stars
Octave	★★★★	**W**	★★★★	★★★★	**NF**★★	**I**	★★	16 Stars
Attack Trees	★★★	**I+Ot**	★	★	**NF** ★★★	**S**	★	9 Stars
USeR	★★★★	**W+Ot**	★★★	★★★★	**NF** ★★★	**S**	★	15 Stars
CLASP	★★	**I+Ot**	★★★★	★	**F**★★ + **NF**★★	**S**	★★	13 Stars
Total Star Count (per question)	27 Stars	NO Stars	32 Stars	27 Stars	23 Stars	NO Stars	15 Stars	**124 Total Star Count**

Figure 9. Best Approaches for Elicitation

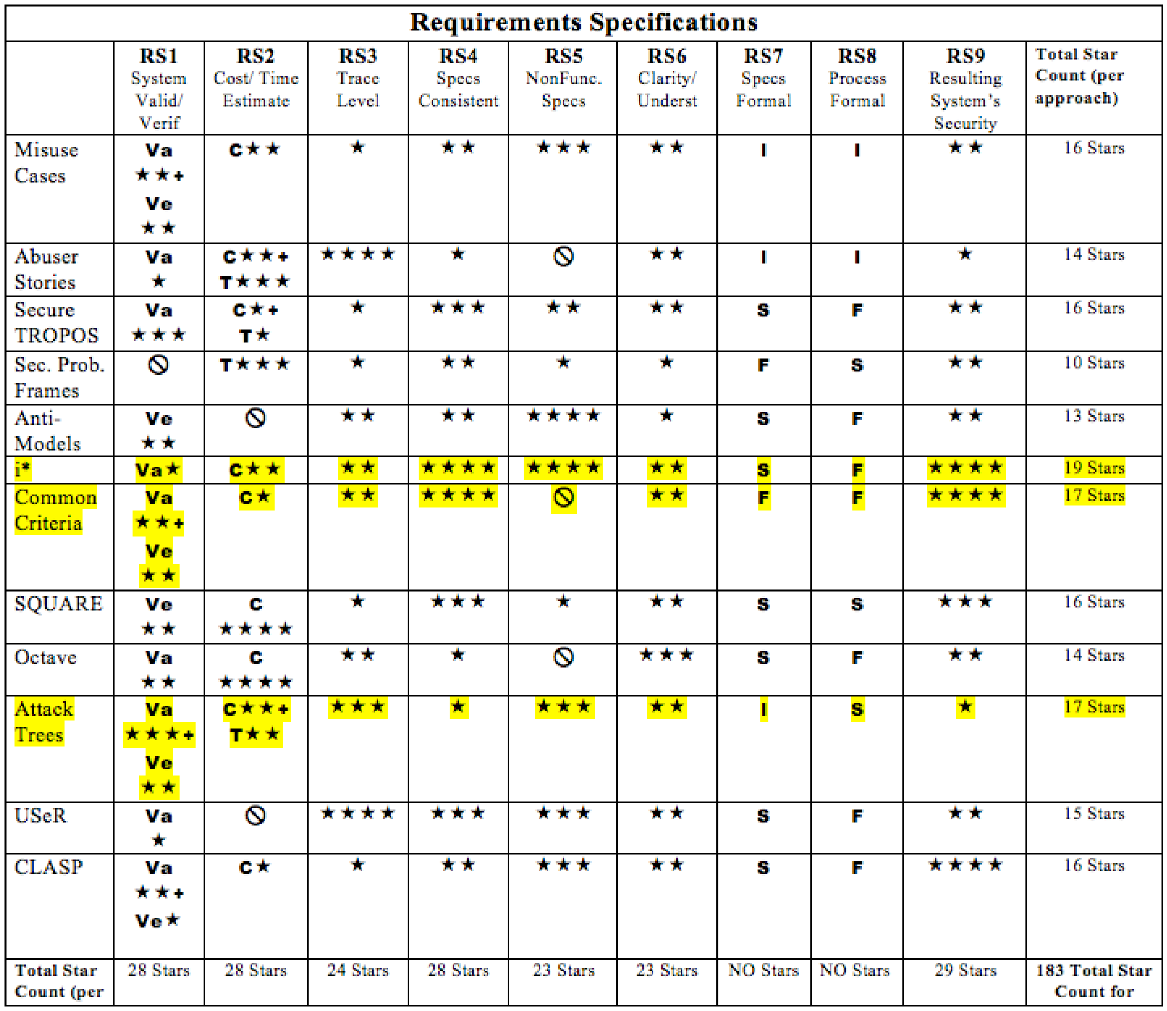
Figure 10 shows the best approaches found for the Analysis phase; in this case we had a total of 3 approaches (Security Problem Frames, Common Criteria, and SQUARE) that came out as best.



Requirements Analysis							
	RA1 Analysis Type	**RA2** Unambig. Resoltn	**RA3** Complete Resoltn	**RA4** Clarity Resoltn	**RA5** Alternative Requs	**RA6** Conflict Resoltn	**Total Star Count (per approach)**
Misuse Cases	I★★★ + E★	★	★★	★	★★	★★★	13 Stars
Abuser Stories	I★	★	★	★★	⊘	⊘	5 Stars
Secure TROPOS	E★★	★	★★	★★	⊘	★★	9 Stars
Sec. Prob. Frames	E★★	★★★	★★★★	★★	★★★	★★	16 Stars
Anti-Models	I★	★★★	★	★★★	★★★	★★	13 Stars
i*	I★★	★	★★	★★★	★★	★★★	13 Stars
Common Criteria	I ★★★★ + E★★★★	★★★★	★★★	★★	★★	★★★	22 Stars
SQUARE	I★★★ + E★★★	★★★	★★	★★★★	★	★★★	20 Stars
Octave	I★★★★	⊘	★★	★★★	★	★	11 Stars
Attack Trees	I★★	★★	⊘	★★	★★★★	★★	12 Stars
USeR	E★★	★★★	★★★★	★★	★★★	★★★	17 Stars
CLASP	I★★★	★	★★	★★	★★★★	★★	16 Stars
Total Star Count (per question)	37 Stars	25 Stars	25 Stars	28 Stars	25 Stars	27 Stars	**167 Total Star Count for Analysis**

Figure 10. Best Approaches for Analysis

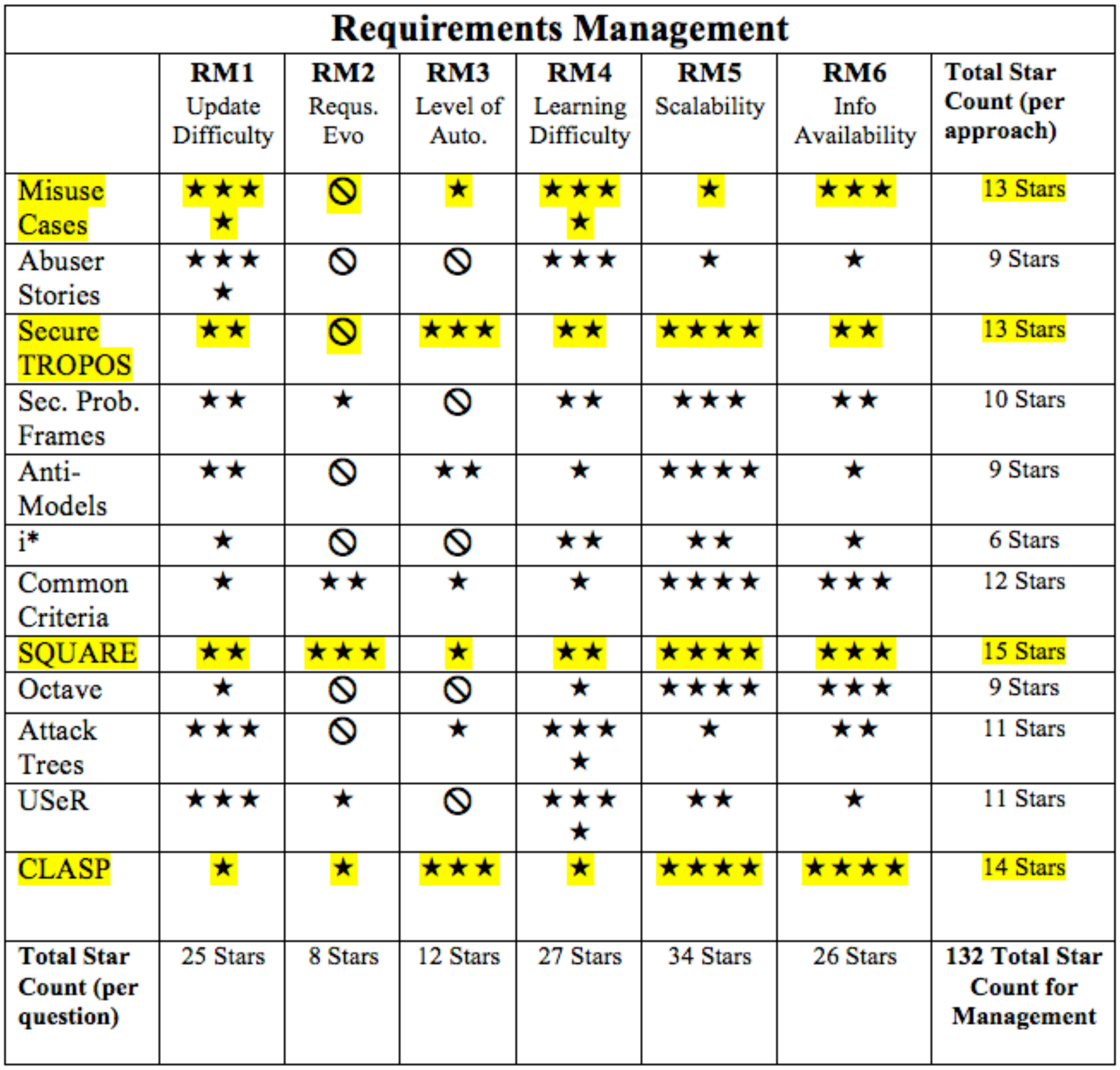
Figure 11 shows the best approaches found for the Specification phase; in this case we had a total of 3 approaches (Misuse Cases, i*, and Common Criteria).



Requirements Specifications										
	RS1 System Valid/ Verif	RS2 Cost/ Time Estimate	RS3 Trace Level	RS4 Specs Consistent	RS5 NonFunc. Specs	RS6 Clarity/ Underst	RS7 Specs Formal	RS8 Process Formal	RS9 Resulting System's Security	Total Star Count (per approach)
Misuse Cases	Va ★★+ Ve ★★	C★★	★	★★	★★★	★★	I	I	★★	16 Stars
Abuser Stories	Va ★	C★★+ T★★★	★★★★	★	⊘	★★	I	I	★	14 Stars
Secure TROPOS	Va ★★★	C★+ T★	★	★★★	★★	★★	S	F	★★	16 Stars
Sec. Prob. Frames	⊘	T★★★	★	★★	★	★	F	S	★★	10 Stars
Anti-Models	Ve ★★	⊘	★★	★★	★★★★	★	S	F	★★	13 Stars
i*	Va★	C★★	★★	★★★★	★★★★	★★	S	F	★★★★	19 Stars
Common Criteria	Va ★★+ Ve ★★	C★	★★	★★★★	⊘	★★	F	F	★★★★	17 Stars
SQUARE	Ve ★★	C ★★★★	★	★★★	★	★★	S	S	★★★	16 Stars
Octave	Va ★★	C ★★★★	★★	★	⊘	★★★	S	F	★★	14 Stars
Attack Trees	Va ★★★+ Ve ★★	C★★+ T★★	★★★	★	★★★	★★	I	S	★	17 Stars
USeR	Va ★	⊘	★★★★	★★★	★★★	★★	S	F	★★	15 Stars
CLASP	Va ★★+ Ve★	C★	★	★★	★★★	★★	S	F	★★★★	16 Stars
Total Star Count (per	28 Stars	28 Stars	24 Stars	28 Stars	23 Stars	23 Stars	NO Stars	NO Stars	29 Stars	183 Total Star Count for

Figure 11. Best Approaches for Specification

Figure 12 shows the best approaches found for the Management phase; in this case we had a total of 4 approaches (Secure Tropos, Common Criteria, SQUARE, and CLASP)



Requirements Management							
	RM1 Update Difficulty	**RM2** Requs. Evo	**RM3** Level of Auto.	**RM4** Learning Difficulty	**RM5** Scalability	**RM6** Info Availability	**Total Star Count (per approach)**
Misuse Cases	★★★★	⊘	★	★★★★	★	★★★	13 Stars
Abuser Stories	★★★★	⊘	⊘	★★★	★	★	9 Stars
Secure TROPOS	★★	⊘	★★★	★★	★★★★	★★	13 Stars
Sec. Prob. Frames	★★	★	⊘	★★	★★★	★★	10 Stars
Anti-Models	★★	⊘	★★	★	★★★★	★	9 Stars
i*	★	⊘	⊘	★★	★★	★	6 Stars
Common Criteria	★	★★	★	★	★★★★	★★★	12 Stars
SQUARE	★★	★★★	★	★★	★★★★	★★★	15 Stars
Octave	★	⊘	⊘	★	★★★★	★★★	9 Stars
Attack Trees	★★★	⊘	★	★★★★	★	★★	11 Stars
USeR	★★★	★	⊘	★★★★	★★	★	11 Stars
CLASP	★	★	★★★	★	★★★★	★★★★	14 Stars
Total Star Count (per question)	25 Stars	8 Stars	12 Stars	27 Stars	34 Stars	26 Stars	**132 Total Star Count for Management**

Figure 12. Best Approaches for Management

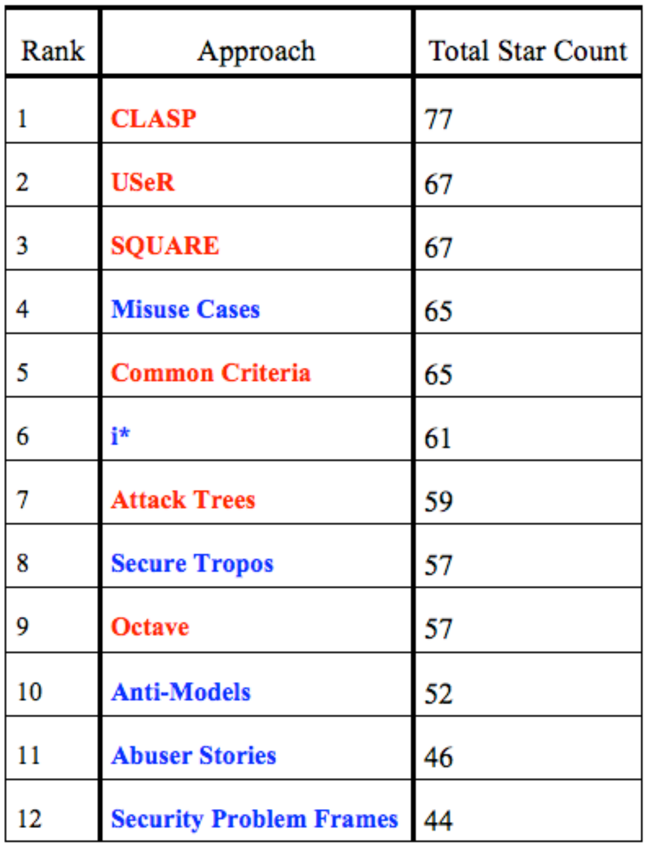
Figure 13 shows the best approaches found for the Later Stages Support phase. This was a very difficult phase to find any decent, let alone best approaches as it will be explained later. We had a total of 3 approaches (misuse cases, Common Criteria, and CLASP) that came out as better ones, each one with 2 "High" responses.

Later Stages Support							
	LSS1 Sec. Requs. Integration	**LSS 2** Constraint Consider	**LSS3** Testing Benefits	**LSS4** Focus for Testing	**LSS5** Reduce Development Effort	**LSS6** Other Requs. Later	**Total Star Count (per approach)**
Misuse Cases	⊘	⊘	★★★	★★★	★★	★	9 Stars
Abuser Stories	⊘	I★★	★	★★	★★	⊘	7 Stars
Secure TROPOS	★★	A★★+ D★★	★★	★	★	★★★	13 Stars
Sec. Prob. Frames	⊘	A★★★★	⊘	★	★★★	⊘	8 Stars
Anti-Models	⊘	⊘	★	★	★★	⊘	4 Stars
i*	★	D★★+ I★★★	★	★	★★★	★	12 Stars
Common Criteria	⊘	D★★ +I★	★	★★	★★★	⊘	9 Stars
SQUARE	⊘	A★	★	★	★★	⊘	5 Stars
Octave	★	⊘	★★	★★	★★	⊘	7 Stars
Attack Trees	⊘	D★★	★★★	★★★★	★	⊘	10 Stars
USeR	★	D★★ +I★	⊘	★	★★★	★	9 Stars
CLASP	★★	A★+D★ ★+ I★+ M★	★★★ ★	★★	★★★★	★	18 Stars
Total Star Count (per question)	7 Stars	29 Stars	19 Stars	21 Stars	28 Stars	7 Stars	**111 Total Star Count for LSS**

Figure 13. Best Approaches for Later Stages Support

7.2 Best Overall Approaches

In order to determine the best approaches throughout the survey, we added the star count per phase of each approach. Figure 14 ranks the 12 approaches surveyed based on their total star count. It is important to mention that according to our results, the top approach, CLASP, offers more support than the rest of the approaches. This higher level of support is not only because CLASP has the highest star count, but also because the difference between CLASP and the runner up, misuse cases, is 5 stars. This 5 star difference in the total star count is the highest difference in star count from among all other approaches. We can therefore deduce that in a way, CLASP offers considereable more support than the other 11 approaches surveyed. On the contrary, security problem frames offer the least amount of support when it comes to security requirements engineering.



Rank	Approach	Total Star Count
1	**CLASP**	77
2	**USeR**	67
3	**SQUARE**	67
4	**Misuse Cases**	65
5	**Common Criteria**	65
6	**i***	61
7	**Attack Trees**	59
8	**Secure Tropos**	57
9	**Octave**	57
10	**Anti-Models**	52
11	**Abuser Stories**	46
12	**Security Problem Frames**	44

Figure 14. Overall Approaches Ranking

8 Results and Observations

Based on the results of the framework application and their comparison, we have formulated certain observations and recommendations that are significant as one analyzes the results of each approach across all of the phases of security requirements engineering. This section discusses 4 specific aspects of our results,

1- General observations about the results
2- Approaches' unexpected strengths
3- Current approaches' weaknesses
4- General recommendations

8.1 General Observations

These are general observations that discuss associations found between various aspects of our survey that are confirmed by our results.

Observation 1- High customer involvement promotes good analysis and good specifications of security requirements

	RE4 Customer Involve	**RS4** Specs Consistent	**RA3** Complete Resoltn	**RA4** Clarity Resoltn	**RA6** Conflict Resoltn	**RS3** Trace Level
Misuse Cases	★★★	★★	★★	★	★★★	★
Abuser Stories	★★★	★	★	★★	⊘	★★★★
Secure TROPOS	★	★★★	★★	★★	★★	★
Sec. Prob. Frames	⊘	★★	★★★★	★★	★★	★
Anti-Models	★★	★★	★	★★★	★★	★★
i*	★★★	★★★	★★	★★★	★★★	★★
Common Criteria	★	★★★	★★★	★★	★★★	★★
SQUARE	★★★★	★★	★★	★★★★	★★★	★
Octave	★★★★	★	★★	★★★	★	★★
Attack Trees	★	★	⊘	★★	★★	★★★
USeR	★★★★	★★★	★★★★	★★	★★★	★★★★
CLASP	★	★★	★★	★★	★★	★

Table 18. Observation 1

Based on the results of the application of the framework, we found that half (6 out of 12) of the approaches have at least a **moderate-high** customer involvement and are represented in yellow in table 18. Based on the responses to the customer involvement level we were able to discover that those approaches that obtained a high response also obtained a high response in a variety of other aspects; thus, high customer involvement is associated to high responses in the aspects below,

1- High customer involvement -> High specification consistency
- A moderate number of approaches with high customer involvement also had a high specification consistency. Out of the 6 approaches with at least a **moderate-high** customer involvement, 4 of them have at least a moderate level of security specifications consistency

2- High customer involvement -> High completeness resolution
- A high number of approaches with high customer involvement also had a high level of completeness resolution in the analysis. Out of the 6 approaches with at least a **moderate-high** level of customer involvement, 5 of them have at least a moderate security requirements completeness resolution level

3- High customer involvement -> High clarity resolution
- A moderate number of approaches with high customer involvement also had a high level of clarity resolution in the analysis. Out of the 6 approaches with at least a **moderate-high** customer involvement, 5 of them have at least a moderate level of security requirements completeness resolution

4- High customer involvement -> High level of requirements conflict resolution
- A moderate number of approaches with high customer involvement also had a high level of requirements conflict resolution in the analysis. Out of the 6 approaches with at least a **moderate-high** customer involvement, 4 of them have high security requirements conflict resolution

5- High customer involvement -> High level of traceability
- A moderate number of approaches with high customer involvement also had a high level of traceability; 4 of them have at least a moderate level of traceability

Observation 2- Formality of the approach's process affects the overall security, consistency, and scalability of the security requirements				
	RS8 Process Formal	**RS9** Resulting System's Security	**RS4** Specs Consistent	**RM5** Scalability
Misuse Cases	I	★★	★★	★
Abuser Stories	I	★	★	★
Secure TROPOS	F	★★	★★★	★★★★
Sec. Prob. Frames	S	★★	★★	★★★
Anti-Models	F	★★	★★	★★★★
i*	F	★★★★	★★★★	★★
Common Criteria	F	★★★★	★★★★	★★★★
SQUARE	S	★★★	★★★	★★★★
Octave	F	★★	★	★★★★
Attack Trees	S	★	★	★
USeR	F	★★	★★★	★★
CLASP	F	★★★★	★★	★★★★

Table 19. Observation 2

We found that a moderate number of approaches (7 out of the 12) surveyed use a formal process for specifying the security requirements; they are represented in yellow in table 3. Based on the approaches that have a formal process for the specification of security requirements we observed that this aspect relates to a high response in a variety of other aspects; thus, a formal process for security requirements correlates to a high level of support in the areas below,

1- Formal Process -> High overall security of the resulting system
- While all of the 7 approaches following a **formal** process to security requirements specification obtained at least a moderate level of overall security of the resulting system, it is important to note that 3 out of these are a high level of overall security. These are the only cases of a high level; specifications specified using a formal process

2- Formal Process -> High specification consistency
- A high number of approaches that use a formal process also had a high level of specification consistency. Out of the 7 approaches with a **formal** specification process, 6 of them have at least a moderate-high specification consistency

3- Formal Process -> High scalability
- A moderate number of approaches that use a formal process also had a high level of scalability. Out of the 7 approaches with a **formal** specification process, 5 of them are highly scalable

Observation 3- Original approaches offer more support for security requirements engineering than derived approaches

	RE Requirements Elicitation	**RA** Requirements Analysis	**RS** Requirements Specification	**RM** Requirements Management	**LSS** Later Stages Support	
Misuse Cases	✓		✓	✓		
Abuser Stories						
Secure TROPOS				✓	✓	
Sec. Prob. Frames						**7**
Anti-Models						
i*			✓		✓	
Common Criteria		✓	✓			
SQUARE		✓		✓		
Octave	✓					**10**
Attack Trees			✓			
USeR	✓	✓				
CLASP				✓	✓	

Table 20. Observation 4 part 1

On table 20, you can see that all (6 out of 6) original approaches are the best approach at least in one phase; only 3 (out of 6) of the derived approaches are the best approach for at least one phase. Furthermore, original approaches come out as best approaches a total of 10 times throughout the 5 phases. In comparison, the derived approaches come out as best approaches only 7 times throughout the 5 phases of the framework. Again, the results prove difficult to determine which set is better that the other, as both are very close to each other.

To further confirm this observation, figure 14 shows how the top three approaches (based on the total amount of stars through out the 5 phases) correspond to original approaches.

8.2 Approaches' Unexpected Strengths

Some surprising results also emerged from our survey; in each phase we found a specific area (question) with an unexpected level of support. Table 23 shows 5 different questions, one per each phase. This is a very positive observation about the strength of the approaches, as they go beyond our expectations when it comes to specific areas of elicitation, analysis, specification, management, and later stages support.

Areas of Unexpected High Support

	RE5 Elicit Other Reqs.	**RA1** Analysis Type	**RS2** Cost/Time Estimate	**RM5** Scalability	**LSS3** Testing Benefits
Misuse Cases	NF★★★	I★★★ E★	C★★	★	★★★
Abuser Stories	⃠	I★	C★★ T★★★	★	★
Secure TROPOS	NF★	E★★	C★ T★	★★★★	★★
Sec. Prob. Frames	⃠	E★★	T★★★	★★★	⃠
Anti-Models	NF★★★	I★	⃠	★★★★	★
i*	NF★	I★★	C★★	★★	★
Common Criteria	NF★	I ★★★★ E★★★★	C★	★★★★	★
SQUARE	F★ NF★	I★★★ E★★★	C★★★★	★★★★	★
Octave	NF★★	I★★★★	C★★★★	★★★★	★★
Attack Trees	NF★★★	I★★	C★★ T★★	★	★★★
USeR	NF★★★	E★★	⃠	★★	⃠
CLASP	F★★ NF★★	I★★★	C★	★★★★	★★★★

Table 23. Unexpected Positive Results

It was refreshing to find in our results that when it comes to elicitation, analysis, and specification there are specific aspects of each that surpass our expectations of support provided.

- Elicitation. We found it interesting that a high number of approaches (10 out of 12) also help elicit other types of requirements besides security. This is surprising because we expected that the approach would be so focused on security that it would not really provide much support for eliciting other types of requirements. While it was not surprising to see that the majority of support was for eliciting other non-functional requirements, a low number of approaches (2 out of 12) provide support for eliciting functional requirements as well.

- Analysis. When it comes to the analysis phase, it was interesting to see that the majority of approaches (8 out of 12) provide a high level of analysis support (either internal, external, or both). While we expected analysis to be an area well covered by approaches, we did not expect to find such high level of support for it.

- Specifications. It was interesting to find out based on our results that a majority of the approaches surveyed (10 out of 12) provide some kind of support for estimating cost, time, or both of the development of the system. This is an added benefit of specifications that we were not expecting the approaches to provide much support for, but it is interesting to see how valuable cost and time estimation is for them.

- Management. We found that there is an unexpectedly high support when it comes to scalability; half of the approaches surveyed are highly scalable.

- Later Stages Support. We did not expect to find any testing support at all; surprisingly there is a moderate level of support when it comes to testing benefits that the approaches surveyed provide.

8.3 Approaches' Weaknesses

This survey uncovers areas that need further development; figure 15 shows the overall support available at each phase based on the surveyed approaches. The figure shows the percentage of current support available for each phase based on the approaches surveyed. This percentage represents the average support for each phase calculated based on the total star count for each approach. The higher the percentage, the more support that is currently being offered for each phase.

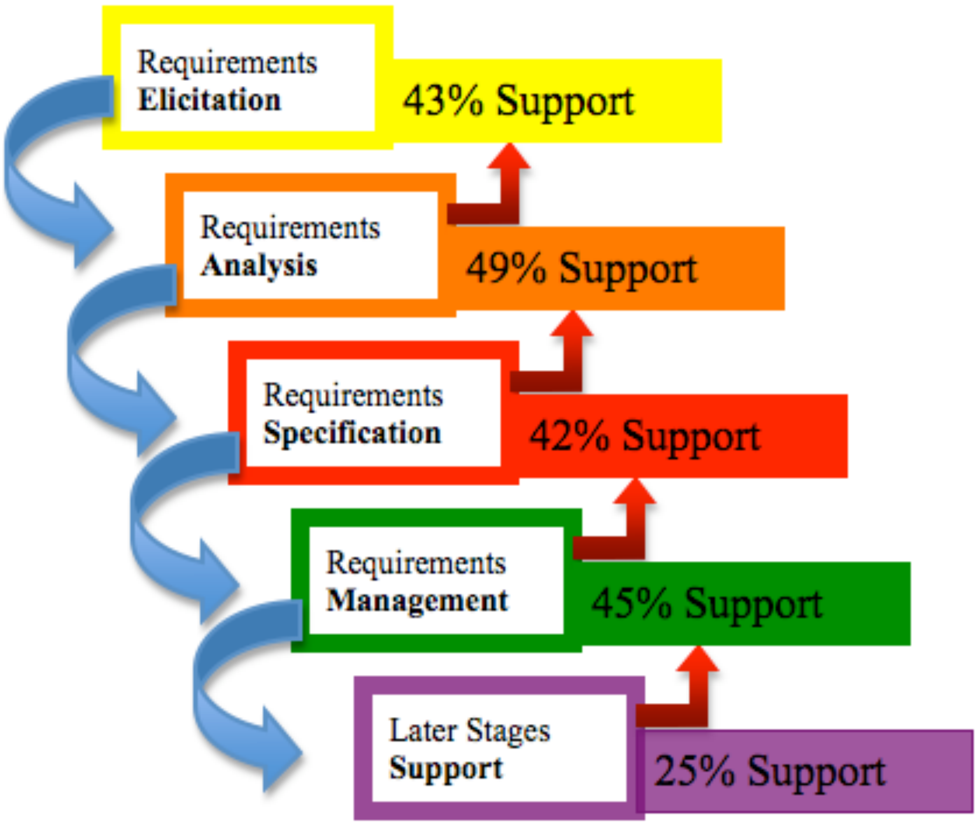


Figure 15. Support per Phase Offered by Surveyed Approaches

The percentages of support for each phase as shown in figure 15 have been calculated based on the ratio between the total star count for each phase obtained by all 12 approaches and the maximum amount of stars available for that phase.

For example,

- Security requirements specification phase has 9 questions in our framework

- Only 7 of these questions have a subjective star rating (None/Low/Moderate/Moderate-High/High) associated with them (therefore there is no star count for two of them)

- 5 of these questions have a maximum number of 4 stars possible. Two questions have a maximum number of 8 stars possible. The maximum number of stars possible is (12*5*4) + (12*2*8) = 432 max stars for specification

- The score for specification is then 183 total star count/ 432 max stars = 42% support

- The standard deviation for the security requirements specification phase is 6.71%

Table 24 shows the calculations for all of the 5 phases explored in the survey; it provides information regarding the total star count for each phase, the maximum amount of stars possible for that phase, and the percentage and standard deviation related to its current support. As explained in the example above, only those questions with star count criteria were considered in the calculations.

Phase	Total Star Count	Max Stars	Percentage of Support	Standard Deviation
Elicitation	124 Stars	288 Stars	43.05%	14.64%
Analysis	167 Stars	336 Stars	49.70%	18.41%
Specification	183 Stars	432 Stars	42.36%	6.71%
Management	132 Stars	288 Stars	45.83%	12.26%
Later Stages Support	111 Stars	432 Stars	25.69%	14.34%

Table 24. Phases Support Calculations

8.3.1 Determining Areas of Weakness

Areas of weakness are determined based on two specific characteristics, percentage of support available and questions with the lowest star count.

8.3.1.1 Percentage of support

As explained above, the higher the percentage, the more support available for that phase; the lower the percentage, the less support that is available. Based on our calculations, we determined that there is a lot more support for the first four phases (elicitation, analysis, specification, and management) than for later stages support. Figure 15 shows the percentage of support for each phase; later stages support, as expected in the introduction, is far more lacking of support.

8.3.1.2 Questions with Lowest Star Count

Another of the advantages of keeping a star count not only horizontally (approach-specific) but also vertically (question-specific) is that it allows us to infer possible areas that lack support based on the questions with the lowest star count. Table 26 shows the 4 questions from our survey that received the lowest star count; based on this we can infer that not only later stages support needs improvement, but also management. Not surprisingly, half of the questions with the lowest star count (2 questions) belong to the later stages support phase; once again verifying that there is a lack of support for security requirements at later stages of development.

Lack of Support for Management and Later Stages				
	RM2 Requs. Evo	**RM3** Level of Auto.	**LSS1** Sec. Requs. Integration	**LSS6** Other Requs. Later
Misuse Cases	🛇	★	🛇	★
Abuser Stories	🛇	🛇	🛇	🛇
Secure TROPOS	🛇	★★★	★★	★★
Sec. Prob. Frames	★	🛇	🛇	🛇
Anti-Models	🛇	★★	🛇	🛇
i*	🛇	🛇	★	★★
Common Criteria	★★	★	🛇	🛇
SQUARE	★★★	★	🛇	🛇
Octave	🛇	🛇	★	🛇
Attack Trees	🛇	★	🛇	🛇
USeR	★	🛇	★	★
CLASP	★	★★★	★★	★
Star Count	**8 Stars**	**12 Stars**	**7 Stars**	**7 Stars**

Table 26. Lack of Support for Last Two Phases

There were a lot of "None" answers for the last two phases of our survey. To put in perspective the results shown in table 26, the average star count per question for the entire survey was 21.7 stars per question. Below we explain the specific areas of need found based on the most lack of support for specific questions,

- No evolution support. The majority of approaches (7 out of 12) provide NO support for security requirements evolution

- No automation support. A moderate number of approaches (5 out of 12) provide NO support for automating any steps/processes of the approach

- No support for security requirements integration. The majority of approaches (8 out of 12) provide NO support for integrating the security requirements in a way that they are useful in later stages of the development

- No support for other types of requirements. The majority of approaches (8 out of 12) provide NO support for other types of requirements besides security in later stages of the development

We can conclude then that both security requirements management and later stages support are phases of security requirements engineering that need to be advanced. With the results of our survey we have narrowed down the areas of need to four of them as presented above; we consider support for integrating and making the security requirements useful at later stages of development the most important one.

8.4 General Recommendations

Once the results have been compared and certain observations were made, we have designed a set of four recommendations that we believe are important when it comes to engineering security requirements. These recommendations can be very helpful for developing security requirements with specific characteristics; how these characteristics can be obtained based on our results is explained below. While we are not arguing that our results definitively prove that certain aspects affect important characteristics of security requirements; we nonetheless have observed that certain approaches that obtain a high level of support in certain questions also receive a high level of support in other aspects. This leads us to recommend that there exist a possibility that improving specific aspects found in our survey, could potentially lead to improvements of specific characteristics of security requirements.

Recommendation1 – How to obtain a high level of traceability					
	RS3 Trace Level	**RE1** Elicit Support	**RE3** Stakehldr ID	**RM1** Update Difficulty	**LSS5** Reduce Development Effort
Misuse Cases	★	★★★★	★★	★★★★	★★
Abuser Stories	★★★★	★★★	★★★★	★★★★	★★
Secure TROPOS	★	★	★★	★★	★
Sec. Prob. Frames	★	⃠	⃠	★★	★★★
Anti-Models	★★	★★	★★★	★★	★★
i*	★★	★★	★★★★	★	★★★
Common Criteria	★★	★	★★	★	★★★
SQUARE	★	★	★★★	★★	★★
Octave	★★	★★★★	★★★★	★	★★
Attack Trees	★★★	★★★	★	★★★	★
USeR	★★★★	★★★★	★★★	★★★	★★★
CLASP	★	★★	★★★★	★	★★★★

Table 27. Recommendation 1

We consider that traceability is extremely important in achieving our objective of integrating security requirements into other stages of the system development. Based on the results of the application of the framework, a small number of the approaches surveyed (3 out of the 12) have at least a moderate-high level of traceability for security requirements. This number is extremely low for those approaches that obtained a high response; there are 2 out of 12 approaches that have a **high** level of traceability of the security specifications, and are represented in yellow in table 27.

Below we show two ways we found, based on our results, for obtaining a high traceability of security requirements. Later on we also discuss the added benefits of having a high level of traceability.

1- High support for elicitation-> High level of traceability
- All of the approaches that have a high support for elicitation have a high level of traceability. Out of the 2 approaches with a **high** level of traceability, both of them have at least a moderate-high level of support for eliciting security requirements

2- High degree of stakeholder identification-> High level of traceability
- All of the approaches that have a high level of stakeholder identification have a high level of traceability. Out of the 2 approaches with a **high** level of traceability, both of them have at least a moderate-high level of support for identifying stakeholders

The added benefits to having high traceability in your security requirements include,

1- High level of traceability-> Easy to update security requirements
- All of the approaches that have a high level of traceability produce security requirements that are easy to update. Out of the 2 approaches with a **high** level of traceability, both of them have at least a moderate-easy level of effort to update the security requirements

2- High level of traceability-> Support to reduce overall development effort
- All of the approaches that have a high level of traceability also help reduce the overall development effort. Out of the 2 approaches with a **high** level of traceability, both of them provide at least a moderate level of support for reducing the overall development effort

Recommendation2 – How to obtain a high level of testing benefits

	LSS3 Testing Benefits	**RE7** System Boundaries	**RA5** Alternative Requs	**RS1** System Valid/ Verif	**RS5** NonFunc. Specs
Misuse Cases	★★★	★★	★★	**Va**★★ **Ve**★★	★★★
Abuser Stories	★	★	🛇	**Va**★★	🛇
Secure TROPOS	★★	★	🛇	**Va**★★★	★★
Sec. Prob. Frames	🛇	🛇	★★★	🛇	★
Anti-Models	★	★★★	★★★	**Ve**★★	★★★★
i*	★	★	★★	**Va**★	★★★
Common Criteria	★	🛇	★★	**Va**★★ **Ve**★★	🛇
SQUARE	★	★	★	**Ve**★★	★
Octave	★★	★★	★	**Va**★★	🛇
Attack Trees	★★★	★	★★★★	**Va**★★★ **Ve**★★	★★★
USeR	🛇	★	★★★	**Va**★	★★★
CLASP	★★★★	★★	★★★★	**Va**★★ **Ve**★	★★★

Table 28. Recommendation 2

Testing benefits is an aspect that we also consider important; based on the results of our framework application we can provide information as to how to obtain them. Based on the results of the application of the framework, we found that a small number of the approaches surveyed (3 out of the 12) have at least a **moderate-high** level of traceability in the security specifications they create. These approaches that provide at least a moderate-high testing benefits are represented in yellow in table 28.

Below we show ways of obtaining a high level of testing benefits from the security requirements,

1- Moderate system boundaries identification -> **High** level of testing benefits
- Most of the approaches that have at least a moderate level of support for system boundaries identification have at least a **moderate-high** level of testing benefits.

2- High elicitation of alternative security requirements -> **High** level of testing benefits
- All of the approaches that have a high support for eliciting alternative/additional security requirements also have a high level of testing benefits. Out of the 2 approaches with **high** level of testing benefits, both of them have a high level of elicitation of alternative security requirements

While testing capabilities are themselves a benefit, there are added benefits associated with high levels of testing support as discovered in our survey,

1- High level of testing benefits -> Support for **both** verification and validation
- All of approaches that have at least a moderate-high level of testing benefits also provide support for using the security requirements specification in the validation and verification of the system.

High level of testing benefits -> **High** level of non-functional requirements specification
- All of approaches that have at least a moderate-high level of testing benefits also provide support for specifying non-functional requirements other than security ones. Out of the 3 approaches with at least **moderate-high** level of testing benefits, all of them provide at least moderate-high support for specifying non-functional requirements besides security

Recommendation 3 – How to obtain a high level of overall security of the resulting system

	RS9 Resulting System's Security	**RE3** Stakehldr ID	**RE4** Customer Involve	**RA2** Unambig. Resoltn	**RM5** Scalability	**LSS 2** Constraint Consider
Misuse Cases	★★	★★	★★★	★	★	⊘
Abuser Stories	★	★★★★	★★★	★	★	I★★
Secure TROPOS	★★	★★	★	★	★★★★	A★★ D★★
Sec. Prob. Frames	★★	⊘	⊘	★★★	★★★	A★★★★
Anti-Models	★★	★★★	★★	★★★	★★★★	⊘
i*	★★★★	★★★★	★★★	★	★★	D★★ I★★★
Common Criteria	★★★★	★★	★	★★★★	★★★★	D★★ I★
SQUARE	★★★	★★★	★★★★	★★★	★★★★	A★

Octave	★★	★★★★	★★★★	🛇	★★★★	🛇
Attack Trees	★	★	★	★★	★	D★★
USeR	★★	★★★	★★★★	★★★	★★	D★★ I★
CLASP	★★★★	★★★★	★	★	★★★★	A★ D★★ I★ M★

Table 29. Recommendation 3

Our last recommendation looks and how to obtain a high level of overall security for your requirements and how this is important. There is a very small number of approaches, 4 out of 12, which we can predict will enable a system to have at least a moderate-high level of security once it is developed. These 4 approaches are represented in yellow in table 29. Below we show ways of obtaining a high level of overall security of your requirements based on the results of our survey,

1- High stakeholder identification -> **High** level of overall security
- Out of the 4 approaches with **high** level of overall security of the specifications, 3 of them have at least a moderate-high level of stakeholder identification

2- High customer involvement -> **High** level of overall security
- Out of the 4 approaches with **high** level of overall security of the specifications, 2 of them have at least a moderate-high level of customer involvement

3- High unambiguity resolution -> **High** level of overall security
- Out of the 4 approaches with **high** level of overall security of the specifications, 2 of them have at least a moderate-high level of unambiguity resolution

While a high level of overall security is one of the main objectives of security requirements engineering, there are added benefits associated with it,

1- High level of overall security **->** **High** level of scalability
- The majority of the approaches that have a high level of overall security also have a high level of scalability. Out of the 4 approaches with **high** level of overall security, 3 of them also provide a high level of scalability

High level of overall security **->** Constraint consideration for design and implementation
- The majority of the approaches that have a high level of overall security also have a high level of constraint consideration for other stages of development. Out of the 4 approaches with **high** level of overall security, 3 of them also provide design and implementation constraint

We believe that these recommendations can be important to developing security requirements. We consider the four aspects covered by our recommendations (clarity and understandability, traceability, testing support, and overall security) vital to a successful set of security requirements. In addition, these four aspects can prove essential in accomplishing our future goals of integrating security requirements with later stages of development, and making them useful.

9 Conclusions

The survey explores the area of security requirements engineering. While this is not a new area, as pointed out by our results there still plenty of work ahead to make it more effective. We believe that our most significant contributions from this survey are as follows,

1. We decompose security requirements engineering into five phases that allow for a more detailed probing of the support offered by each approach at each specific phase.
2. Our framework explores a variety of aspects that are important to securely engineering requirements.
3. Through our results, we are able to verify some requirements engineering best practices.
4. Our survey unveils some difficulties that exist today, preventing us from characterizing how to increase integration support for security requirements at later stages of the development lifecycle and determining if approaches derived from existing ones are better than those created specifically for security
5. We have determined that there are two major areas that need improvement when it comes to security requirements engineering; management and later stages support.
6. Based on our results, we provide a variety of recommendations in order to help developers produce better security requirements.

Below we summarize our survey and provide a glimpse into how the information that was obtained in this survey will be used in our future research.

9.1 Conclusions Summary

The survey explored the stage of security requirements engineering from the point of view of decomposing it into 5 sub phases. We designed an evaluation framework that probed the support offered by current approaches to elicitation, analysis, specification, management, and later stages support of security requirements. 12 different approaches were surveyed; these were separated into approaches derived from others in order to address security and approaches designed specifically for security.

General observations as well as strengths and weaknesses of current approaches were identified; general recommendations were also made based on them. We were able to determine that both security requirements management and later stages support are phases of security requirements engineering that need to be advanced. With the results of our survey we have narrowed down the areas of need to four of them, security requirements integration, automation, evolution, and support for other types of requirements. We consider support for integrating and making the security requirements useful at later stages of development the most important one. The findings of this survey will be of tremendous help in addressing this need.

Looking back at our survey expectations described in the introduction, our results showed that,

1. Most of the approaches surveyed did better in the initial 4 phases of security requirements engineering (elicitation, analysis, specification, management) but there was a lack of support for integrating them and making them useful at later stages of development.

2. Based on our results it was difficult to determine if approaches created specifically for security requirements did better than those that have been adapted from existing ones to address security. Therefore we cannot declare either one the definite winner.

3. While there was no approach that provided a lot more support than the rest throughout the 5 phases, CLASP did significantly better than the other 11 approaches surveyed based on the total star count.

We can then conclude that integration of security requirements into later stages of development is a key aspect in advancing security requirements engineering. Our survey showed that there is still a lot of work needed to support this integration. In addition, based on our observations, we can conclude that improvements to specific aspects of security requirements engineering should help ultimately improve their integration. For example, we consider clarity and understandability, traceability, testing support, and overall security to be key aspects in improving security requirements integration into later stages of developments. The recommendations made for obtaining these four aspects identified other characteristics that affect them, like traceability is affected by stakeholder identification for example. Improving these characteristics will ultimately improve security requirements integration.

9.2 Research Objectives and Future Directions

The research to come will be shaped by the results obtained in this survey; it will aim to increase the support available for integrating security requirements at later stages of development.

We determined that support for integration of security requirements into later stages of development is one of the most important needs in security requirements engineering today. We have decided to address this need by proposing a new approach that will provide developers with guidance and support for tracing the security requirements into architecture, design, implementation, and ultimately testing. Our main goal is to not only provide traceability support for the security requirements, but to make them a vital and useful part of each stage.

Based on the results of our framework application, we consider that there is one approach that can be extended in order to make the security requirements it produces useful at later stages of development; this approach is CLASP. As mentioned in the results section, CLASP obtained the highest star count and we believe that it will be an appropriate starting point for extending security requirements. In the case that CLASP proves to be not suited for our objective, we will explore two other approaches that were also surveyed. These two approaches are SQUARE and USeR. SQUARE and USeR are also considered as possible candidates because they proved to have a good potential when it comes to their usability as well as their traceability. Instead of developing our own approach for security requirements engineering we are looking to extend either CLASP, SQUARE, or USeR in order to make their security requirements more traceable and ultimately useful at the testing phase of development. We will take each approach and decide which one is better suited for our needs, but in the case that neither of them prove to be effective enough then we will consider developing our own approach to security requirements engineering. The results of the survey will be extremely useful in determining not only what the new approach should include, but also what aspects we need to steer clear from.

Our proposed future directions seem promising in addressing the lack of support for security requirements at later stages of development; we want to ultimately extend what we learn from security to help address a variety of other non-functional requirements.

10 References

[Abr98] Abrams, M.: Application of the Protection Profile to Define Requirements for a Telecommunications Services Contract. IEEE Software, 15(2). 1998

[Aka97] Akao, Y.: QFD: Past, present and future. Transactions of the Third International Symposium on Quality Function Deployment. 1997

[Alb02] Alberts, C., Dorofee, A.: Managing Information Security Risks: The OCTAVE (SM) Approach. Addison-Wesley. 2002

[Alb05] Alberts, C. et. al.: Introduction to the OCTAVE Approach. CERT Coordination Center. www.cert.org/octave/approach_intro.pdf

[Alb99] Alberts, C., Behrens, S., Pethia, R., Wilson, W.: Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE-SM) Framework, Version 1.0 (CMU/SEI-99-TR-017, ADA 367718). Software Engineering Institute, Carnegie Mellon University. 1999

[Ale02] Alexander, I.: Initial Industrial Experience of Misuse Cases in Trade-Off Analysis. Proceedings of IEEE Joint International Requirements Engineering Conference. 2002

[Ale03] Alexander, I.: Misuse Cases Help to Elicit Non-Functional Requirements. Computing and Control Engineering. 2003

[Bos06] Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K., Kruchten, P.: Extending XP Practices to Support Security Requirements Engineering. Proceedings of Workshop on Software Engineering for Secure Systems (SESS). 2006

[Bre04] Bresciani, P., Giorgini, P, Giunchiglia, F, Mylopoulos, J., Perini, A.: TROPOS: An Agent Oriented Software Development Methodology. Journal of Autonomous Agents and Multi-Agent Systems. 2004

[CCIB99] Common Criteria Implementation Board: Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements. ISO/IEC 15408-1. 1999

[Cha02] Chan, L.K. and Wu, M.L.: Quality Function Deployment: A Comprehensive Review of its Concepts and Methods. Quality Engineering, 15(1). 2002

[ChD04] Chen, P., Dean, M., Ojoko-Adams, D., Osman, H., Lopez, L., Xie, N.: Systems Quality Requirements Engineering (SQUARE) Methodology: Case Study on Asset Management System. (CMU/SEI-2004-SR-015). Software Engineering Institute, Carnegie Mellon University. 2004

[ChF05] Chivers, H. and Fletcher, M.: Applying Security Design Analysis to a service based system. Software: Practice and Experience, vol. 35 no. 9. 2005

[Chr92] Christel, M. and Kang, K.: Issues in Requirements Elicitation (CMU/SEI-92-TR-012, ADA258932). Software Engineering Institute, Carnegie Mellon University. 1992

[Coh95] Cohen, L.: Quality Function Deployment: How to Make QFD Work for You. Addison-Wesley. 1995

[CSO06] Codesecurely.org: Security Requirements Engineering. 2006 http://www.codesecurely.org/Wiki/view.aspx/Security_Requirements_Engineering

[Dav1] Davis, A.: Software Requirements: Analysis and Specification. Prentice Hall. 1990

[Dor90] Dorfman, M.: Tutorial: System and Software Requirements Engineering. IEEE Computer Society Press. 1990

[Eas00] Easterbrook, S., Nuseibeh, B.: Requirements Engineering: A Roadmap. Proceedings of the International Conference on Software Engineering. 2000

[ESI96] European Software Institute: European User Survey Analysis. Report USV_EUR 2.1, ESPITI Project. 1996

[Fir03] Firesmith, D.: Engineering Security Requirements. Journal of Object Technology. 2003

[Fir07] Firesmith, D.: Engineering Safety and Security Related Requirements for Software Intensive Systems. ICSE Companion. 2007

[GiM05] Giorgini, P., Massacci, F., Zannone, N.: Security and Trust Requirements Engineering. Foundations of Security Analysis and Design III - Tutorial Lectures. 2005

[Gio06] Giorgini, P., Mouratidis, H., Zannone, Z.: Modelling Security and Trust with Secure Tropos. Integrating Security and Software Engineering: Advances and Future Vision. 2006

[Gor05] Gordon, D., Stehney II, G., Wattas, N., Yu, E.: Quality Requirements Engineering (SQUARE): Case Study on Asset Management System, Phase II (CMU/SEI-2005-SR005). Software Engineering Institute, Carnegie Mellon University. 2005

[HaH06] Hallberg, N. Hallberg, J.: The Usage-Centric Security Requirements Engineering (USeR) Method. Information Assurance Workshop. 2006

[HaH07] Hatebur, D., Heisel, M., Schmidt, H.: A Pattern System for Security Requirements Engineering. Proceedings of the International Conference on Availability, Reliability and Security (AReS). 2007

[Hal04] Haley, C., Laney, R., Nuseibeh, B.: Deriving Security Requirements from Crosscutting Threat Descriptions. AOSD. 2004

[HaL07] Haley, C. Laney, R., Moffett, J., Nuseibeh, B.: Security Requirements Engineering: A Framework for Representation and Analysis. IEEE Transactions on Software Engineering. 2007

[Hat05] Hatebur, D., Heisel, M.: Problem Frames and Architectures for Security Problems. SAFECOMP. 2005

[Hat06] Hatebur, D., Heisel, M., Schmidt, H.: Security Engineering Using Problem Frames. Proceedings of the International Conference on Emerging Trends in Information and Communication Security (ETRICS). 2006.

[Hat07] Hatebur, D., Heisel, M., Schmidt, H.: A Security Engineering Process based on Patterns. DEXA Workshops. 2007

[Hog04] Höglund, G., McGraw, G.: Exploiting Software : How to Break Code. Addison Wesley Professional. 2004

[Hop04] Hope, P., McGraw, G., Anton, A.: Misuse and Abuse Cases: Getting Past the Positive. Security & Privacy, IEEE Volume 02, Issue 3. 2004

[IAT07] Information Assurance Technology Analysis Center (IATAC) and Data Analysis Center for Software (DACS): Software Security Assurance: State-of-the-Art-Report. 2007

[IEEE98] IEEE: IEEE Recommended Practice for Software Requirements Specifications. 1998. http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=15571

[ISO99] ISO/IEC.: Information Technology - Security Techniques - Evaluation Criteria for IT Security - Part 1: Introduction and General Model. ISO/IEC. International Standard 15408-1. 1999

[Jac01] Jackson, M.: Problem Frames. Analyzing and Structuring Software Development Problems. Addison Wesley. 2001

[Jac92] Jacobson, I. et al.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley. 1992

[Kam05] Kam, S.: Integrating the Common Criteria Into the Software Engineering Lifecycle. IDEAS'05. 2005

[Kul00] Kulak, D., Guiney, E.: Use Cases: Requirements in Context. ACM Press. 2000

[Lam04] Lamsweerde, A.: Elaborating Security Requirements by Construction of Intentional Anti-Models. 26th International Conference on Software Engineering (ICSE'04). 2004

[Lew02] Lewis, R.: Design for Security Up Front. 2002 http://articles.techrepublic.com.com/5100-10878-1059545.html

[Lin97] Linger, R., Mead, N., Lipson, H.: Requirements Definition for Survivable Network Systems. Software Engineering Institute, Carnegie Mellon University. 1997

[Liu03] Liu, L., Yu, E., Mylopoulos, J.: Security and Privacy Requirements Analysis within a Social Setting In. Proceedings of the International Conference on Requirements Engineering (RE). 2003

[Lou89] Loucopoulos, P., and Champion. R.E.M.: Knowledge-Based Support for Requirements Engineering. Information and Software Technology. 1989

[Luc04] Bastos, L., Brelaz de Castro, J.: Systematic Integration Between Requirements and Architecture. SELMAS. 2004

[Lut07] Lutz, R., Patterson-Hine, A., Nelson, S., Frost, C., Tal, D., Harris, R.: Using Obstacle Analysis to Identify Contingency Requirements on an Unpiloted Aerial Vehicle. Requirements Engineering Journal. Vol. 12. No. 1. 2007

[McG03] McGraw, G.: Software Security: Thought Leadership in Information Security. Cigital Software Security Workshop. 2003

[Mea05] Mead, N., Hough, E., Stehney II, T.: Security Quality Requirements (SQUARE) Methodology. (CMU/SEI-2005-TR-009). Software Engineering Institute, Carnegie Mellon University. 2005

[Mea06] Mead, N.: Security Requirements Engineering. Carnegie Mellon University. 1996. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/requirements/243.html

[Mea07] Mead, N.: How To Compare the Security Quality Requirements Engineering (SQUARE) Method with Other Methods. Technical Note, CMU/SEI. 2007

[Mead04] Mead, N.: Requirements Elicitation and Analysis Processes for Safety & Security Requirements. Proceedings of Fourth International Workshop on Requirements for High Assurance Systems. 2004

[MeF06] Mellado, D., Fernández-Medina, E., Piattini, M.: A Comparative Study of

Proposals for Establishing Security Requirements for the Development of Secure Information Systems. Proceedings International Conference on Computational Science and its Applications (ICCSA). 2006

[MeF07] Mellado, D., Fernández-Medina, E., Piattini, M.: A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems. Computer Standards & Interfaces, vol 29. 2007

[MGS03] Mouratidis, H., Giorgini, P., Schumacher, M., Manson, M.: Security Patterns for Agent Systems. Proceedings of the Eight European Conference on Pattern Languages of Programs (EuroPLoP). 2003

[Mof03] Moffett, J. D. and Nuseibeh, B.A.: A Framework for Security Requirements Engineering. Report YCS 368, Department of Computer Science, University of York. 2003

[MoG03] Mouratidis, H., Giorgini, P., Manson, G.: Modelling Secure Multiagent Systems. Proceedings of the Second International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS). 2003

[MoH04] Moffett, J. Haley, C., Nuseibeh, B.: Core Security Requirements Artefacts. Technical Report 2004/23. Department of Computing, The Open University. 2004

[Moo01] Moore, A. et al.: Attack Modeling for Information Security and Survivability. Technical Note CMU/SEI-2001-TN-001. Software Engineering Institute, Carnegie Mellon University. 2001

[Mou03] Mouratidis, H., Giorgini, P., Manson G.: Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems. Proceedings of the 15th Conference on Advance Information Systems (CAiSE). 2003

[Mou06] Mouratidis, H. and Giorgini, P.: Secure Tropos: Dealing effectively with Security Requirements in the development of Multiagent Systems. Safety and Security in Multiagent Systems. LNCS, Springer-Verlag, 2006

[NY01] New York State Office for Technology.: Requirements Analysis. 2001

[Olt01] Olthoff, K.: Observations on Security Requirements Engineering. Symposium on Requirements Engineering for Information Security. 2001

[Pau93] Paulk, M., Weber, C., Garcia, S., Chrissis, M., Bush, M.: Key Practices of the Capability Maturity Model, Version 1.1. Software Engineering Institute, Carnegie Mellon University. CMU/SEI-93-TR-25. 1993

[Pet05] Peeters, J.: Agile Security Requirements Engineering. Symposium Requirements Engineering Information Security, 2005
www.sreis.org/SREIS_05_Program/short26_peeters.pdf

[Pet07] Peeters, J. and Dyson, P.: Cost-Effective Security. IEEE Security & Privacy. 2007

[Pie01] Piessens, F., De Decker, B., De Win, B.: Developing secure software. A survey and classification of common software vulnerabilities. IICIS. 2001

[Red04] Redwine, S. et al.: Processes to Produce Secure Software: Towards More Secure Software. National Cyber Security Summit. 2004

[Ric07] Caralli, R., Stevens, J., Young, L., Wilson, W.: Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process. Technical Report CMU/SEI-2007-TR-012. Software Engineering Institute, Carnegie Mellon University
http://iac.dtic.mil/iatac/download/security.pdf

[Rom07] Romero-Mariona, J., Ziv, H., Richardson, D.: Toward Hybrid Requirements-based and Architecture-based Testing. Proceedings of The Role of Software Architecture for Testing and Analysis (ROSATEA). 2007

[Rom90] Rombach, H.: Software Specifications: A Framework. IEEE Tutorial on Standards, Guidelines, and Examples: Systems and Software Requirements Engineering. ISBN 0-8186-8922-6. 1990

[Rum94] Rumbaugh, J.: Getting Started: Using use cases to capture requirements. Journal of Object-Oriented Programming. 1994

[Rze89] Rzepka, W.: A Requirements Engineering Testbed: Concept, Status, and First Results. In Bruce D. Shriver (editor), Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. IEEE Computer Society. 1989

[Sch00] Schneier, B.: Secrets and Lies: Digital Security in a Networked World. John Wiley & Sons. 2000

[SEI91] Software Engineering Institute Requirements Engineering Project: Requirements Engineering and Analysis Workshop Proceedings. Technical Report CMU/SEI-91-TR-30 or ESD-TR-91-30, Software Engineering Institute. 1991

[Sin03] Sindre, G., Firesmith, D., Opdahl, A.: A Reuse-Based Approach to Determining Security Requirements. Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ). 2003

[SSI05] Secure Software Inc.: The CLASP Application Security Process. 2005

[STEP91] Software Test & Evaluation Panel (STEP), Requirements Definition Implementation Team: Operational Requirements for Automated Capabilities, Draft Pamphlet (Draft PAM). 1991

[Sto01] Stoneburner, G., Hayden, C., & Feringa, A.: Engineering Principles for Information Technology Security (A Baseline for Achieving Security). Computer Security Division, Information Technology Laboratory National Institute of Standards and Technology. 2001

[Tar95] Tarr, C., Peaty, S.: Using CLASP to Assess Perimeter Security. Proceedings Institute of Electrical and Electronics Engineers 29th Annual International Carnahan Conference. 1995

[Tun08] Tundel, Jaatun, Moland.: Security Requirements for the Rest of Us: A Survey. IEEE Software. 2008

[Vet02] Vetterling, M. et al.: Secure Systems Development Based on the Common Criteria: The PalME Project. Foundations of Software Engineering (SIGSOFT). 2002

[Vie01] Viega, J., McGraw, G.: Building Secure Software: How to Avoid Security Problems the Right Way. 1st ed. Addison-Wesley. 2001

[Vie05] Viega, J.: Building Security Requirements with CLASP. Proceedings of the Workshop on Software Engineering for Secure Systems (SESS). 2005

[War06] Ware, M.: Using Common Criteria to Elicit Security Requirements with Use Cases. Proceedings of the IEEE SoutheastCon. 2006

[Wei98] Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P.: Scenario Usage in System Development: A Report on Current Practice. IEEE Software. 1998

[Wel03] Welch, D., Lathrop, S.: A Survey of 802.11a Wireless Security Threats and Security Mechanisms. ITOC Technical Report 2003-101 to the Army G6. 2003

[Whi01] Whitmore, J.: A Method for Designing Secure Solutions. IBM Systems Journal. Volume: 40. Issue: 3. 2001

[Zah90] Zahniser, Richard A.: How to Speed Development with Group Sessions. IEEE Software. 1990

[Zav97] Zave, P., Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology, 6(1). ACM Press. 1997

[Zuk89] Zucconi, L.: Techniques and Experiences Capturing Requirements for Several Real-Time Applications. ACM SIGSOFT Software Engineering Notes. 1989